Information Sciences 485 (2019) 230–247

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Fast construction of correcting ensembles for legacy Artificial Intelligence systems: Algorithms and a case study☆

Ivan Yu. Tyukin[a,b,c,*], Alexander N. Gorban[a,c], Stephen Green[a], Danil Prokhorov[d]

[a] University of Leicester, Department of Mathematics, University Road, Leicester, LE1 7RH, UK
[b] Department of Automation and Control Processes, St. Petersburg State University of Electrical Engineering, Prof. Popova str. 5, Saint-Petersburg, 197376, Russian Federation
[c] Lobachevsky State University of Nizhnij Novgorod, Prospekt Gagarina 23, Nizhnij Novgorod,603022, Russian Federation
[d] Toyota R & D, Ann-Arbor, MI, USA

ARTICLE INFO

Article history:
Received 29 May 2018
Revised 28 October 2018
Accepted 10 November 2018
Available online 7 February 2019

MSC:
68T05
68T42
97R40
60F05

Keywords:
Neural ensembles
Stochastic separation theorems
Perceptron
Artificial intelligence
Measure concentration

ABSTRACT

This paper presents a new approach for constructing simple and computationally efficient improvements of generic Artificial Intelligence (AI) systems, including Multilayer and Deep Learning neural networks. The improvements are small network ensembles added to the existing AI architectures. Theoretical foundations of the approach are based on stochastic separation theorems and the ideas of the concentration of measure. We show that, subject to mild technical assumptions on statistical properties of internal signals in the original AI system, the approach enables fast removal of the AI's errors with probability close to one on the datasets which may be exponentially large in dimension. The approach is illustrated with numerical examples and a case study of digits recognition in American Sign Language.

1. Introduction

Artificial Intelligence (AI) systems have risen dramatically from being the subject of niche academic and practical interests to a commonly accepted and widely-spread facet of modern life. Industrial giants such as Google, Amazon, IBM, and Microsoft offer a broad range of AI-based services, including intelligent image and sound processing and recognition.

Deep Learning and related computational technologies [8,27] are currently perceived as state-of-the art tools for producing various AI systems. In visual recognition tasks, these systems deliver unprecedented accuracy [40] at reasonable computational costs [29]. Despite these advances, several fundamental challenges hinder further progress of these technologies.

All data-driven AI systems make mistakes, regardless of how well an AI system is trained. Some examples of these have already received global public attention [6,15]. Mistakes may arise due to uncertainty in empirical data, data misrepresenta-

☆ this paper belongs to the special issue "RANN" edited by Prof. W. Pedrycz.

* Corresponding author at: University of Leicester, Department of Mathematics, University Road, Leicester, LE1 7RH, UK.
E-mail addresses: I.Tyukin@le.ac.uk (I.Yu. Tyukin), a.n.gorban@le.ac.uk (A.N. Gorban), slg46@le.ac.uk (S. Green).

https://doi.org/10.1016/j.ins.2018.11.057
0020-0255/

tion, and imprecise or inaccurate training. Conventional approaches to reducing spurious errors include altering training data and improving design procedures [33,35,37,49], transfer learning [10,36,48] and privileged learning [44]. These approaches invoke extensive training procedures. The training itself, whilst eradicating some errors, may introduce new errors by the very nature of the steps involved (e.g. randomized sampling of mini-batches, randomized training sets etc).

In this work, we propose a training approach and a technology whereby errors of the original legacy AI are removed, with high probability, via incorporation of small *neuronal ensembles* and their cascades into the original AI's decision-making. Ensembles methods and classifiers' cascades (see e.g. [16,17,38,45] and references therein) constitute a well-known framework for improving performance of classifiers. Here we demonstrate that geometry of high-dimensional spaces, in agreement with "blessing of dimensionality" [12,21,31] enables efficient construction of AI error correctors and their ensembles with guaranteed performance bounds.

The technology bears similarity with neurogenesis deep learning [13], classical cascade correlation [14], greedy approximation [5], and randomized methods for training neural networks [41], including deep stochastic configuration networks [46,47]. The proposed technology, however, does not require computationally expensive training or pre-conditioning, and in some instances can be set up as a non-iterative procedure. In the latter case the technology's computational complexity scales at most linearly with the size of the training set.

At the core of the technology is the concentration of measure phenomenon [18,19,24,25,34], a view on the problem of learning in AI systems that is stemming from conventional probabilistic settings [11,43], and stochastic separation theorems [20,21,23]. Main building blocks of the technology are simple threshold, perceptron-type [39] classifiers. The original AI system, however, needs not be a classifier itself. We show that, subject to mild assumptions on statistical properties of signals in the original AI system, small neuronal ensembles consisting of cascades of simple linear classifiers are an efficient tool for learning away spurious and systematic errors. These cascades can be used for learning new tasks too.

The paper is organized as follows: Section 2 contains a formal statement of the problem, Section 3 presents main results: namely, the algorithms for constructing correcting ensembles (Section 3.2) preceded by their mathematical justification (Section 3.1). Section 4 presents a case study on American Sign Language recognition illustrating the concept, and Section 5 concludes the paper. Proofs of theorems and other statements as well as additional technical results are provided in Appendix A and Appendix B.

Notation

The following notational agreements are used throughout the paper:

- $\mathbb{R}$ denotes the field of real numbers;
- $\mathbb{N}$ is the set of natural numbers;
- $\mathbb{R}^n$ stands for the n-dimensional real space; unless stated otherwise symbol n is reserved to denote dimension of the underlying linear space;
- Let $\boldsymbol{x}, \boldsymbol{y}$ be elements of $\mathbb{R}^n$, then $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \sum_{i=1}^{n} x_i y_i$ is their inner product;
- Let $\boldsymbol{x} \in \mathbb{R}^n$, then $\|\boldsymbol{x}\|$ is the Euclidean norm of $\boldsymbol{x}$: $\|\boldsymbol{x}\| = \sqrt{\langle \boldsymbol{x}, \boldsymbol{x} \rangle}$;
- $B_n(R)$ denotes a n-ball of radius R centered at 0: $B_n(R) = \{\boldsymbol{x} \in \mathbb{R}^n | \ \|\boldsymbol{x}\| \leq R\}$;
- If $\mathcal{S}$ is a finite set then $|\mathcal{S}|$ stands for its cardinality;
- $\mathcal{V}(\Xi)$ is the Lebesgue volume of $\Xi \subset \mathbb{R}^n$;
- If X is a random variable then $E[X]$ is its expected value.

2. Problem statement

In what follows we suppose that an AI system is an operator mapping elements of its input set, $\mathcal{U}$, to the set of outputs, $\mathcal{Q}$. Examples of inputs $\boldsymbol{u} \in \mathcal{U}$ are images, temporal or spatiotemporal signals, and the outputs $\boldsymbol{q} \in \mathcal{Q}$ correspond to labels, classes, or some quantitative characteristics of the inputs. Inputs $\boldsymbol{u}$, outputs $\boldsymbol{q}$, and internal variables $\boldsymbol{z} \in \mathcal{Z}$ of the system represent the system's state. The state itself may not be available for observation but some of its variables or relations may be accessed. In other words, we assume that there is a process which assigns an element of $\boldsymbol{x} \in \mathbb{R}^n$ to the triple $(\boldsymbol{u}, \boldsymbol{z}, \boldsymbol{q})$. A diagram illustrating the setup for a generic AI system is shown in Fig. 1.

In addition to the core AI system we consider an add-on ensemble mapping samples $\boldsymbol{x}$ into auxiliary signals $\boldsymbol{s} \in \mathcal{S}$ (shown as *improvement signals* in Fig. 1) which are then used to improve performance of the original core AI. An example of such an improvement signal could be an indication that the current state of the core AI system represented by $\boldsymbol{x}$ corresponds to an erroneous decision of the core AI. At the integration stage, this information will alter the overall response.

With regards to the operations f_i performed at the individual nodes (circles in the diagram on Fig. 1), we will focus on the following relevant class

$$f_i(\boldsymbol{x}) = f(\langle \boldsymbol{w}_i, \boldsymbol{x} \rangle - c_i), \tag{1}$$

where the function $f : \mathbb{R} \to \mathbb{R}$ is piece-wise continuous, $\boldsymbol{w}_i \in \mathbb{R}^n$, and $c_i \in \mathbb{R}$. This class of functions is broad enough to enable the ensemble to function as a universal approximation device (e.g. by choosing the function f in the class satisfying conditions discussed in [5]) as well as to fit into a wide range of common AI architectures including decision trees, multilayer

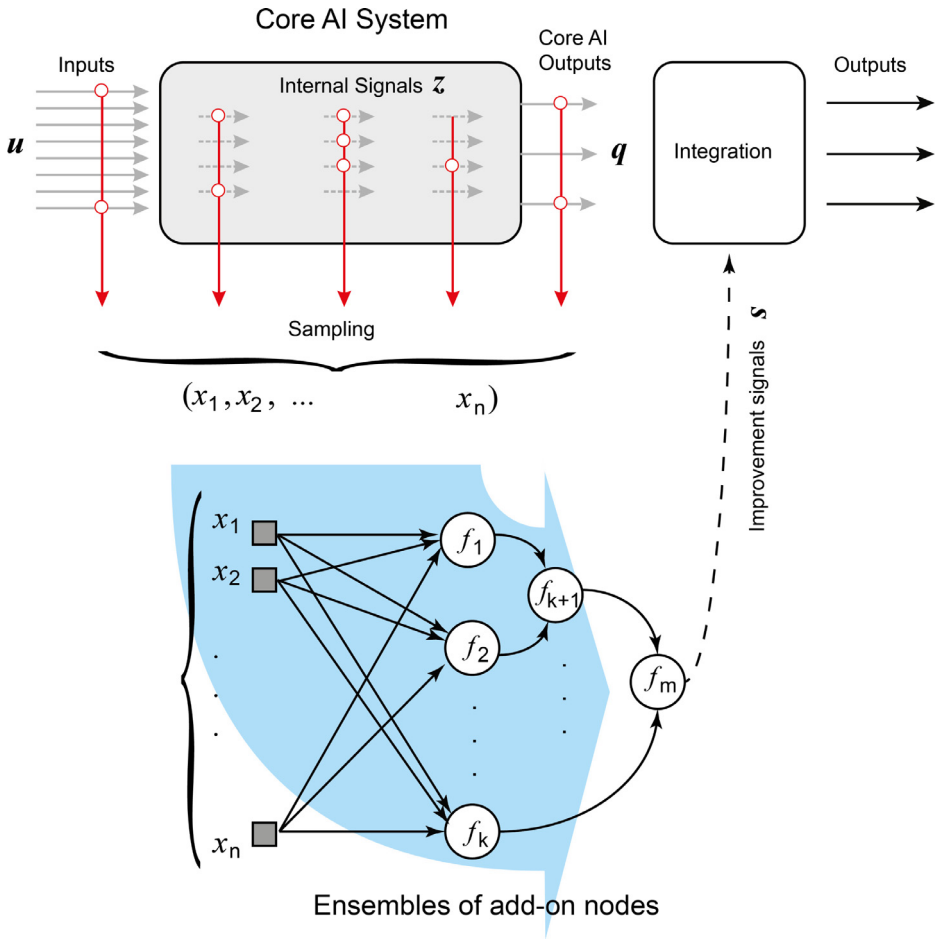

Fig. 1. Functional diagram of correcting neuronal ensembles for a generic Artificial Intelligence system.

neural networks and perceptions, and deep learning convolutional neuronal networks. Additional insight into "isolation" properties of elements (1) in comparison with more natural choices such as e.g. ellipsoids or balls (cf. [1]) is provided in Appendix B.

Over a relevant period of time, the AI system generates a finite but large set of measurements $\boldsymbol{x}_i$. This set is assessed by an external supervisor and is partitioned into the union of the sets $\mathcal{M}$ and $\mathcal{Y}$:

$$\mathcal{M} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M\}, \;\; \mathcal{Y} = \{\boldsymbol{x}_{M+1}, \ldots, \boldsymbol{x}_{M+k}\}.$$

The set $\mathcal{M}$ may contain measurements corresponding to expected operation of the AI, whereas elements from $\mathcal{Y}$ constitute core AI's performance singularities. These singularities may be both desired (related e.g. to "important" inputs $\boldsymbol{u}$) and undesired (related e.g. to errors of the AI). The function of the ensemble is to respond to these singularities selectively by producing improvement signals $\boldsymbol{s}$ in response to elements from the set $\mathcal{Y}$.

A straightforward approach to derive such ensembles is to use standard optimization facilities such as e.g. error back-propagation and its various modifications, or use other iterative procedures similar to greedy approximation [5], stochastic configuration networks [47] (cf. [13]). This approach, however, requires computational resources and time that may not necessarily be available. Even in the simplest case of a single-element add-on system that is to separate the entire set $\mathcal{X}$ from $\mathcal{Y}$, determining the best possible solution, in the form of e.g. support vector machines [43], is non-trivial computationally. Indeed, theoretical worst-case estimates of computational complexity for determining parameters of the single-element system are of the order $O((M+k)^3)$ [9].

The question, therefore is: if there exists a computationally efficient procedure for determining ensembles or add-on networks such that

1) They are able to "improve" performance of a generic AI system with guaranteed probability, and
2) They consist of elements (1)?

Preferably, computational complexity of the procedure is to be linear or even sub-linear in $M + k$.

In the next sections we show that, subject to some mild technical assumptions on the way the sets $\mathcal{M}$ and $\mathcal{Y}$ are generated, a family of simple algorithms with the required characteristics can indeed be derived. These algorithms are motivated by Stochastic Separation Theorems [20,21,23]. For consistency, adapted version of these theorems are presented and discussed in Section 3.1. The algorithms themselves are presented and discussed in Section 3.2.

3. Main results

3.1. Mathematical preliminaries

Following standard assumptions (see e.g. [11,43]), we suppose that all $\boldsymbol{x}$ are generated in accordance with some distribution, and the actual measurements $\boldsymbol{x}_i$ are samples from this distribution. For simplicity, we adopt a traditional setting in which all such samples be identically and independently distributed (i.i.d.) [11]. With regards to the elements of $\boldsymbol{x}_i$, the following technical condition is assumed:

Assumption 1. Elements $\boldsymbol{x}_i$ are random i.i.d. vectors drawn from a product measure distribution:

A1) their x_{ij}th components are independent and bounded random variables X_j: $-1 \le X_j \le 1,\ j = 1, \ldots, n$,
A2) $E[X_j] = 0$, and $E[X_j^2] = \sigma_j^2$.

The distribution itself, however, is supposed to be unknown. Let

$$R_0^2 = \sum_{i=1}^{n} \sigma_i^2 > 0.$$

Then the following result holds (cf. [20]).

Theorem 1. *Let $\boldsymbol{x}_i \in \mathcal{M} \cup \mathcal{Y}$ be i.i.d. random points from the product distribution satisfying Assumption 1, $0 < \delta < 1$, $0 < \varepsilon < 1$ and $R_0 > 0$. Then*

1) *for any i,*

$$\begin{aligned} &P\left(1 - \varepsilon \le \frac{\|\boldsymbol{x}_i\|^2}{R_0^2} \le 1 + \varepsilon\right) \ge 1 - 2\exp\left(-\frac{2R_0^4\varepsilon^2}{n}\right);\\ &P\left(\frac{\|\boldsymbol{x}_i\|^2}{R_0^2} \ge 1 - \varepsilon\right) \ge 1 - \exp\left(-\frac{2R_0^4\varepsilon^2}{n}\right); \end{aligned} \tag{2}$$

2) *for any $i, j, i \ne j$,*

$$\begin{aligned} &P\left(\left\langle \frac{\boldsymbol{x}_i}{R_0}, \frac{\boldsymbol{x}_j}{R_0}\right\rangle < \delta\right) \ge 1 - \exp\left(-\frac{R_0^4\delta^2}{2n}\right);\\ &P\left(\left\langle \frac{\boldsymbol{x}_i}{R_0}, \frac{\boldsymbol{x}_j}{R_0}\right\rangle > -\delta\right) \ge 1 - \exp\left(-\frac{R_0^4\delta^2}{2n}\right); \end{aligned} \tag{3}$$

3) *for any given $\boldsymbol{y} \in [-1, 1]^n$ and any i*

$$P\left(\left\langle \frac{\boldsymbol{x}_i}{R_0}, \frac{\boldsymbol{y}}{R_0}\right\rangle < \delta\right) \ge 1 - \exp\left(-\frac{R_0^4\delta^2}{2n}\right). \tag{4}$$

Proof of Theorem 1 as well as those of other technical statements are provided in Appendix A.

Remark 1. Notice that if $\sigma_i^2 > 0$ then $R_0^2 > n\min_i\{\sigma_i^2\}$. Hence the r.h.s. of (2)–(4) become exponentially close to 1 for n large enough.

The following Theorem is now immediate.

Theorem 2 (1-Element separation)**.** *Let elements of the set $\mathcal{M} \cup \mathcal{Y}$ be i.i.d. random points from the product distribution satisfying Assumption 1, $0 < \varepsilon < 1$, and $R_0 > 0$. Consider $\boldsymbol{x}_{M+1} \in \mathcal{Y}$ and let*

$$\ell_1(\boldsymbol{x}) = \left\langle \frac{\boldsymbol{x}}{R_0}, \frac{\boldsymbol{x}_{M+1}}{R_0}\right\rangle,\ h_1(\boldsymbol{x}) = \ell_1(\boldsymbol{x}) - 1 + \varepsilon. \tag{5}$$

Then

$$\begin{aligned} &P(h_1(\boldsymbol{x}_{M+1}) \ge 0 \text{ and } h_1(\boldsymbol{x}_i) < 0 \text{ for all } \boldsymbol{x}_i \in \mathcal{M})\\ &\ge 1 - \exp\left(-\frac{2R_0^4\varepsilon^2}{n}\right) - M\exp\left(-\frac{R_0^4(1-\varepsilon)^2}{2n}\right). \end{aligned} \tag{6}$$

(The proof is provided in Appendix A.)

Remark 2. Theorem 2 not only establishes the fact that the set $\mathcal{M}$ can be separated away from $\mathcal{Y}$ by a linear functional with reasonably high probability. It also specifies the separating hyperplane, (5), and provides an estimate from below of the probability of such an event, (6). The estimate, as a function of n, approaches 1 exponentially fast. Note that the result is intrinsically related to the work [32] on quasiorthogonal dimension of Euclidian spaces.

Let us now move to the case when the set $\mathcal{Y}$ contains more than one element. Theorem 3 below summarizes the result.

Theorem 3 (*k-Element separation. Case 1).* *Let elements of the set $\mathcal{M} \cup \mathcal{Y}$ be i.i.d. random points from the product distribution satisfying Assumption 1, and $R_0 > 0$. Let, additionally,*

$$\left\langle \frac{\boldsymbol{x}_i}{R_0}, \frac{\boldsymbol{x}_j}{R_0} \right\rangle \leq \beta \tag{7}$$

for all $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{Y}$, $\boldsymbol{x}_i \neq \boldsymbol{x}_j$. Pick

$$0 < \varepsilon < 1, \;\; 1 - \varepsilon + \beta(k-1) > 0,$$

and consider

$$\begin{aligned} \ell_k(\boldsymbol{x}) &= \left\langle \frac{\boldsymbol{x}}{R_0}, \frac{\bar{\boldsymbol{x}}}{R_0} \right\rangle, \;\; \bar{\boldsymbol{x}} = \frac{1}{k}\sum_{i=1}^{k} \boldsymbol{x}_{M+i}, \\ h_k(\boldsymbol{x}) &= \ell_k(\boldsymbol{x}) - \frac{1 - \varepsilon + \beta(k-1)}{k}. \end{aligned} \tag{8}$$

Then

$$\begin{aligned} &P\big(h_k(\boldsymbol{x}_j) \geq 0 \;\&\; h_k(\boldsymbol{x}_i) < 0 \text{ for all } \boldsymbol{x}_i \in \mathcal{M}, \;\; \boldsymbol{x}_j \in \mathcal{Y}\big) \\ &\geq 1 - k\exp\left(-\frac{2R_0^4\varepsilon^2}{n}\right) - M\exp\left(-\frac{R_0^4(1 - \varepsilon + \beta(k-1))^2}{2k^2 n}\right). \end{aligned} \tag{9}$$

(The proof is provided in Appendix A.)

The value of β in estimate (7) may not necessarily be available a-priori. If this is the case then the following corollaries from Theorems 2 and 3 may be invoked.

Corollary 1 (*k-Element separation. Case 1).* *Let elements of the set $\mathcal{M} \cup \mathcal{Y}$ be i.i.d. random points from the product distribution satisfying Assumption 1, and $R_0 > 0$. Pick*

$$0 < \delta, \;\; 0 < \varepsilon < 1, \;\; 1 - \varepsilon - \delta(k-1) > 0,$$

and let

$$\begin{aligned} \ell_k(\boldsymbol{x}) &= \left\langle \frac{\boldsymbol{x}}{R_0}, \frac{\bar{\boldsymbol{x}}}{R_0} \right\rangle, \;\; \bar{\boldsymbol{x}} = \frac{1}{k}\sum_{i=1}^{k} \boldsymbol{x}_{M+i}, \\ h_k(\boldsymbol{x}) &= \ell_k(\boldsymbol{x}) - \frac{1 - \varepsilon - \delta(k-1)}{k}. \end{aligned} \tag{10}$$

Then

$$\begin{aligned} P(h_k(\boldsymbol{x}_j) \geq 0 \,\&\, h_k(\boldsymbol{x}_i) < 0 \text{ for all } \boldsymbol{x}_i \in \mathcal{M}, \;\; \boldsymbol{x}_j \in \mathcal{Y}) \\ \geq \; 1 - k\exp\left(-\frac{2R_0^4\varepsilon^2}{n}\right) - \frac{k(k-1)}{2}\exp\left(-\frac{R_0^4\delta^2}{2n}\right) \\ - M\exp\left(-\frac{R_0^4(1 - \varepsilon - \delta(k-1))^2}{2k^2 n}\right). \end{aligned} \tag{11}$$

Corollary 2 (*k-Element separation. Case 2).* *Let elements of the set $\mathcal{M} \cup \mathcal{Y}$ be i.i.d. random points from the product distribution satisfying Assumption 1, $0 < \varepsilon < 1$, $0 < \mu < 1 - \varepsilon$ and $R_0 > 0$. Pick $\boldsymbol{x}_j \in \mathcal{Y}$ and consider*

$$\begin{aligned} \ell_k(\boldsymbol{x}) &= \left\langle \frac{\boldsymbol{x}}{R_0}, \frac{\boldsymbol{x}_j}{R_0} \right\rangle, \;\; h_k(\boldsymbol{x}) = \ell_k(\boldsymbol{x}) - 1 + \varepsilon + \mu, \\ \Omega &= \left\{ \boldsymbol{x} \in \mathbb{R}^n \;\middle|\; \left\langle \frac{\boldsymbol{x}_j}{R_0}, \frac{\boldsymbol{x}_j - \boldsymbol{x}}{R_0} \right\rangle \leq \mu \right\} \end{aligned} \tag{12}$$

Then

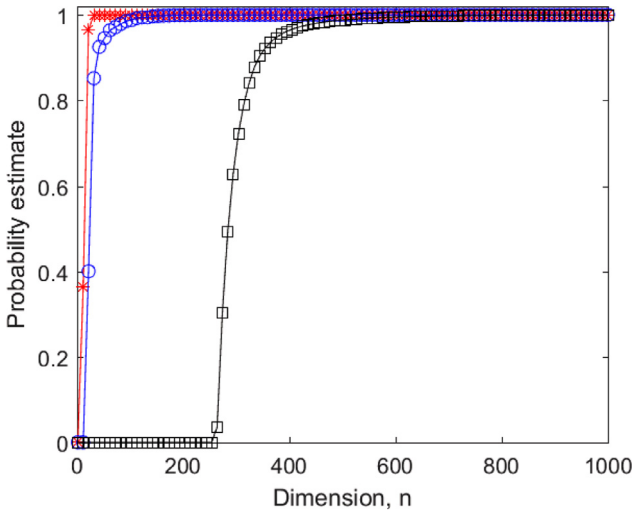

Fig. 2. The probability (frequency) that an element of the set $\mathcal{M}$ (formed by random i.i.d. vectors drawn from the n-cube $[-1,1]^n$) is separable from the rest by hyperplanes (5) (and their variants) as a function of n; $M = |\mathcal{M}| = 10^4$. Red stars correspond to empirical frequencies of the events A_i: $\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle < \|\boldsymbol{x}_i\|^2$ for all $\boldsymbol{x}_j \in \mathcal{M}$, $\boldsymbol{x}_i \neq \boldsymbol{x}_j$. Blue circles show frequencies of the events B_i: $\|\boldsymbol{x}_i\|^2/R_0^2 \geq 1-\varepsilon$ and $\langle \boldsymbol{x}_i/R_0, \boldsymbol{x}_j/R_0 \rangle < 1-\varepsilon$ for all $\boldsymbol{x}_j \in \mathcal{M}$, $\boldsymbol{x}_i \neq \boldsymbol{x}_j$, $\varepsilon = 0.2$. Black squares show the right-hand side of (14) for the same value of $\varepsilon = 0.2$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

$$P(h_k(\boldsymbol{x}) \geq 0 \text{ and } h_k(\boldsymbol{x}_i) < 0 \text{ for all } \boldsymbol{x}_i \in \mathcal{M}, \boldsymbol{x} \in \Omega) \geq 1 - k\exp\left(-\frac{2R_0^4\varepsilon^2}{n}\right) - M\exp\left(-\frac{R_0^4(1-\varepsilon-\mu)^2}{2n}\right). \tag{13}$$

Proofs of the corollaries are provided in Appendix A.

Theorems 1–3 and Corollaries 1, 2 suggest that simple elements (1) with f being mere threshold elements

$$f(s) = \mathrm{Step}(s) = \begin{cases} 1, & s \geq 0 \\ 0, & s < 0 \end{cases},$$

posses remarkable selectivity. For example, according to Theorem 2, the element

$$f(\langle \boldsymbol{x}, \boldsymbol{w} \rangle - c), \quad \boldsymbol{w} = \frac{\boldsymbol{x}_{M+1}}{R_0^2}, \quad c = \frac{\|\boldsymbol{x}_{M+1}\|^2}{R_0^2}$$

assigns "1" to $\boldsymbol{x}_{M+1}$ and "0" to all $\boldsymbol{x}_i \in \mathcal{M}$ with probability that is exponentially (in n) close to 1. To illustrate this point, consider a simple test case with a set comprising of 10^4 i.i.d. samples from the (uniform) product distribution in $[-1,1]^n$. For this distribution, $R_0 = \sqrt{n/3}$, $|\mathcal{M}| = M = 9999$, and estimate (6) becomes:

$$P(h_1(\boldsymbol{x}_{M+1}) \geq 0 \text{ and } h_1(\boldsymbol{x}_i) < 0 \text{ for all } \boldsymbol{x}_i \in \mathcal{M}) \geq 1 - \exp\left(-\frac{2}{9}n\varepsilon^2\right) - M\exp\left(-\frac{1}{18}n(1-\varepsilon)^2\right). \tag{14}$$

The right-hand side rapidly approaches 1 as n grows. The estimate, however, could be rather conservative as is illustrated with Fig. 2.

A somewhat more relaxed approach could be to allow a small margin of error by determining a hyperplane separating the set $\mathcal{Y}$ from "nearly all" elements $\boldsymbol{x} \in \mathcal{M}$. An estimate of success for the latter case follows from Lemma 1 below

Lemma 1. *Let elements of the set $\mathcal{M}$ be i.i.d. random points, and let*

$$\ell(\boldsymbol{x}) = \langle \boldsymbol{x}, \boldsymbol{w} \rangle, \quad h(\boldsymbol{x}) = \ell(\boldsymbol{x}) - c$$

and $0 \leq p_* \leq 1$ be such that

$$P(h(\boldsymbol{x}_i) \geq 0) \leq p_*$$

for an arbitrary element $\boldsymbol{x}_i \in \mathcal{M}$. Then

$$P(h(\boldsymbol{x}) \geq 0 \text{ for at most } m \text{ elements } \boldsymbol{x} \in \mathcal{M}) \geq (1-p_*)^M \exp\left(\frac{(M-m+1)p_*}{1-p_*}\right)\left(1 - \frac{1}{m!}\left(\frac{(M-m+1)p_*}{(1-p_*)}\right)^m\right).$$

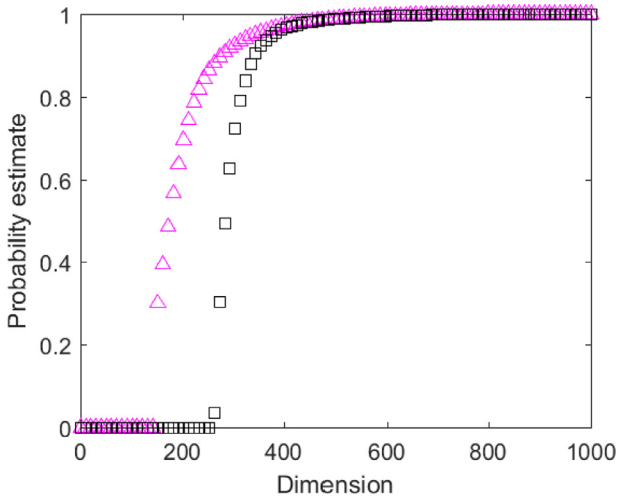

Fig. 3. Probability estimates (14) (black squares) and (15) (magenta triangles) as functions of n for the set $\mathcal{M}$ formed by random i.i.d. vectors drawn from the n-cube $[-1,1]^n$ and $\varepsilon = 0.2$. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

(The proof is provided in Appendix A.)
According to Lemma 1 and Theorem 1,

$$\begin{aligned} &P(h_1(\boldsymbol{x}_{M+1}) \geq 0 \text{ and } h_1(\boldsymbol{x}_i) \geq 0 \text{ for at most } n\ \boldsymbol{x}_i \in \mathcal{M}) \\ &\geq (1-p_*)^M \exp\left(\frac{(M-n+1)p_*}{1-p_*}\right)\left(1-\frac{1}{n!}\left(\frac{(M-n+1)p_*}{(1-p_*)}\right)^n\right) \\ &\quad -\exp\left(-\frac{2R_0^4\varepsilon^2}{n}\right),\ \ p_* = \exp\left(-\frac{R_0^4(1-\varepsilon)^2}{2n}\right). \end{aligned} \tag{15}$$

Observe, however, that $n+1$ random points (in general position) are linearly separable with probability 1. Thus, with probability 1, spurious n points can be separated away from $\boldsymbol{x}_{M+1}$ by a second hyperplane. Fig. 3 compares separation probability estimate for such a pair with that of for h_1 (Theorem 2, (6)).

3.2. Fast AI up-training algorithms

Theorem 3 as well as Corollaries 1,2 and Lemma 1 motivate a simple computational framework for construction of networks and cascades of elements (1). Following the setup presented in Section 2, recall that the data is sampled from the core AI system and is partitioned into the sets $\mathcal{M}$ and $\mathcal{Y}$. The latter set $\mathcal{Y}$ corresponds to singular events which are to be picked by the cascade. Their union, $\mathcal{S} = \mathcal{M} \cup \mathcal{Y}$, is the entire data that is available for the cascade construction.

We are now ready to proceed with the algorithms for fast construction of the correcting ensembles. The first algorithm is a recursion of which each iteration is a multi-step process. The algorithm is provided below.

Algorithm 1 (Correcting Ensembles)**.** Initialization: set $\mathcal{M}^1 = \mathcal{M}$, $\mathcal{Y}^1 = \mathcal{Y}$, and $\mathcal{S}^1 = \mathcal{S}$, define a list or a model of *correcting actions* – formalized alternations of the core AI in response to an error/error type.

Input to the ith iteration: Sets $\mathcal{M}^i$, $\mathcal{Y}^i$, and $\mathcal{S}^i = \mathcal{M}^i \cup \mathcal{Y}^i$; the number of clusters, p, and the value of filtering threshold, θ.

1. *Centering.* All current data available is centered. The centered sets are denoted as $\mathcal{S}_c^i$ and $\mathcal{Y}_c^i$ and are formed by subtracting the mean $\overline{\boldsymbol{x}}(\mathcal{S}^i)$ from the elements of $\mathcal{S}^i$ and $\mathcal{Y}^i$, respectively:

$$\mathcal{S}_c^i = \{\boldsymbol{x} \in \mathbb{R}^n | \boldsymbol{x} = \xi - \overline{\boldsymbol{x}}(\mathcal{S}^i),\ \xi \in \mathcal{S}^i\}$$

$$\mathcal{Y}_c^i = \{\boldsymbol{x} \in \mathbb{R}^n | \boldsymbol{x} = \xi - \overline{\boldsymbol{x}}(\mathcal{S}^i),\ \xi \in \mathcal{Y}^i\}.$$

2. *Regularization.* The covariance matrix, $\mathrm{Cov}(\mathcal{S}^i)$, of $\mathcal{S}^i$ is calculated along with the corresponding eigenvalues and eigenvectors. New regularized sets $\mathcal{S}_r$, $\mathcal{Y}_r$ are produced as follows. All eigenvectors $h_1, h_2, \ldots, h_m$ that correspond to the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_m$ which are above a given threshold are combined into a single matrix $H = (h_1|h_2|\cdots|h_m)$. The threshold could be chosen on the basis of the Kaiser–Guttman test [30] or otherwise (e.g. by keeping the ratio of the maximal to the minimal in the set of retained eigenvalues within a given bound). The new sets are defined as:

$$\mathcal{S}_r^i = \{\boldsymbol{x} \in \mathbb{R}^m | \boldsymbol{x} = H^T\xi,\ \xi \in \mathcal{S}_c^i\}$$

$$\mathcal{Y}_r^i = \{\boldsymbol{x} \in \mathbb{R}^m | \boldsymbol{x} = H^T\xi,\ \xi \in \mathcal{Y}_c^i\}.$$

3. *Whitening.* The two sets then undergo a whitening coordinate transformation ensuring that the covariance matrix of the transformed data is the identity matrix:

$$\mathcal{S}_w^i = \{\boldsymbol{x} \in \mathbb{R}^m | \boldsymbol{x} = W\xi,\ \xi \in \mathcal{S}_r^i,\ W = \mathrm{Cov}(\mathcal{S}_r^i)^{-\frac{1}{2}}\}$$

$$\mathcal{Y}_w^i = \{\boldsymbol{x} \in \mathbb{R}^m | \boldsymbol{x} = W\xi,\ \xi \in \mathcal{Y}_r^i,\ W = \mathrm{Cov}(\mathcal{S}_r^i)^{-\frac{1}{2}}\}.$$

4. *Projection (optional).* Project elements of $\mathcal{S}_w^i$, $\mathcal{Y}_w^i$ onto the unit sphere by scaling them to the unit length: $\boldsymbol{x} \mapsto \varphi(\boldsymbol{x})$, $\varphi(\boldsymbol{x}) = \boldsymbol{x}/\|\boldsymbol{x}\|$.
5. *Training: Clustering.* The set $\mathcal{Y}_w^i$ (the set of errors) is then partitioned into p clusters $\mathcal{Y}_{w,1}^i, \ldots, \mathcal{Y}_{w,p}^i$ that's elements are pairwise positively correlated.
6. *Training: Nodes creation and aggregation.* For each $\mathcal{Y}_{w,j}^i$, $j = 1, \ldots, p$ and its complement $\mathcal{S}_w^i \setminus \mathcal{Y}_{w,j}^i$ we construct the following separating hyperplanes:

$$h_j(\boldsymbol{x}) = \ell_j(\boldsymbol{x}) - c_j,$$
$$\ell_j(\boldsymbol{x}) = \left\langle \frac{\boldsymbol{w}_j}{\|\boldsymbol{w}_j\|}, \boldsymbol{x} \right\rangle,\ c_j = \min_{\xi \in \mathcal{Y}_{w,j}^i} \left\langle \frac{\boldsymbol{w}_j}{\|\boldsymbol{w}_j\|}, \xi \right\rangle$$
$$\boldsymbol{w}_j = (\mathrm{Cov}(\mathcal{S}_w^i \setminus \mathcal{Y}_{w,j}^i) + \mathrm{Cov}(\mathcal{Y}_{w,j}^i))^{-1}(\overline{\boldsymbol{x}}(\mathcal{Y}_{w,j}^i) - \overline{\boldsymbol{x}}(\mathcal{S}_w^i \setminus \mathcal{Y}_{w,j}^i)).$$

Retain only those hyperplanes for which $c_j > \theta$. For each retained hyperplane, create a corresponding element (1)

$$f_j(\boldsymbol{x}) = f\left(\left\langle WH^T(\boldsymbol{x} - \overline{\boldsymbol{x}}(\mathcal{S}^i)), \frac{\boldsymbol{w}_j}{\|\boldsymbol{w}_j\|}\right\rangle - c_j\right), \tag{16}$$

with f being a function that satisfies: $f(s) > 0$ for all $s \geq 0$ and $f(s) \leq 0$ for all $s < 0$, and add it to the ensemble. If optional step 4 was used then the functional definition of f_j becomes:

$$f_j(\boldsymbol{x}) = f\left(\left\langle \varphi\left(WH^T(\boldsymbol{x} - \overline{\boldsymbol{x}}(\mathcal{S}^i))\right), \frac{\boldsymbol{w}_j}{\|\boldsymbol{w}_j\|}\right\rangle - c_j\right), \tag{17}$$

where φ is the mapping implementing projection onto the unit sphere.

7. *Integration/Deployment.* Any $\boldsymbol{x}$ that is generated by the original core AI is put through the ensemble of $f_j(\cdot)$. If for some $\boldsymbol{x}$ any of the values of $f_j(\boldsymbol{x}) > 0$ then a *correcting action* is performed on the core AI. The action is dependent on the purpose of correction as well as on the problem at hand. It could include label swapping, signalling an alarm, ignoring/not reporting etc. The combined system becomes new core AI.
8. *Testing.* Assess performance of new core AI on a relevant data set. If needed, generate new sets $\mathcal{M}^{i+1}$, $\mathcal{Y}^{i+1}$, $\mathcal{S}^{i+1}$ (with possibly different error types and definitions), and repeat the procedure.

Steps 1–3 of the algorithm are standard pre-processing routines. In the context of Theorems 1–3, step 1 aims to ensure that the first part of A2) in Assumption 1 holds, and step 2, in addition to regularization, results in that all components of $\boldsymbol{x}_i$ are uncorrelated (a necessary condition for part A1) of Assumption 1 to hold). Whitening transformation, step 3, normalizes the data in the sense that $\sigma_i^2 = 1$ for all i relevant. Step 4 (optional) is introduced to account for data irregularities and clustering that may negatively affect data separability. An illustration of potential utility of this step is shown in Fig. 4. Note that individual components of the data vectors may no longer satisfy the independence assumption. Nevertheless, if the data is reasonably distributed, different versions of separation theorems may apply to this case too [21,23]. In Section 4 we illustrate the effect of this step in an example application.

Step 5, clustering, is motivated by Theorem 3 and Corollaries 1, 2 suggesting that the rate of success in isolating multiple points from the background set $\mathcal{M}$ increases when these multiple points are positively correlated or are spatially close to each other.

Step 6 is a version of (8), albeit with a different normalization and a slight perturbation of weights. The choice of this particular normalization is motivated by the experiments shown in Fig. 2. As for the choice of weights, these hyperplanes are standard Fisher discriminants. Yet, they are not far from (8). In view of the previous steps, $\mathrm{Cov}(\mathcal{S}_w^i \setminus \mathcal{Y}_{w,j}^i) + \mathrm{Cov}(\mathcal{Y}_{w,j}^i)$ is diagonally dominated and close to the identity matrix. When $|\mathcal{S}_w^i| \gg |\mathcal{Y}_{w,j}^i|$, term $\overline{\boldsymbol{x}}(\mathcal{S}_w^i \setminus \mathcal{Y}_{w,j}^i)$ is nearly zero, and hence $\boldsymbol{w}_j$ is approximately at the centroid of the cluster. Filtering of the nodes is necessitated by the concentration of measure effects as exemplified by Theorem 1.

Functions f in step 7 can be implemented using a range of ReLU, threshold, $\tanh(\cdot)$, sigmoidal functions etc.; these are available in the majority of standard core AI systems.

Computational complexity of each step in the recursion, except for step 5, is at most $M + k$. Complexity of the clustering step may generally be superpolynomial (worst case) as is e.g. the case for standard k-means clustering [3]. If, however,

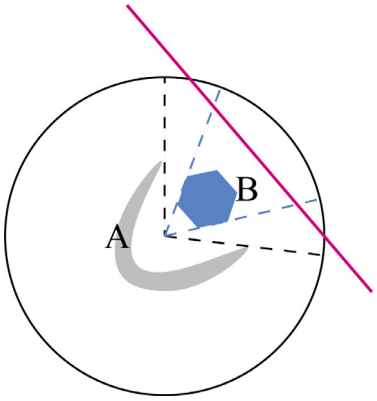

Fig. 4. Linear separability via projection onto a sphere. Sets A (shown as a shaded gray domain) and B (depicted as a blue filled hexagon) are not linearly separable in the original coordinates. Their projections onto the sphere, however, are separable (the separating hyperplane is shown in red). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

sub-optimal solutions are accepted [26] then the complexity of this step scales linearly with $M+k$. Further speed-ups are possible through randomized procedures such as e.g. in k-means++ [2].

Algorithm 1 is based primarily on the intuition and rationale stemming from Theorems 2, 3 and their corollaries. It can be modified to take advantage of the possibility offered by Lemma 1 and the subsequent discussion. The modification is that each node added in step 6 is assessed and "corrected" by an additional hyperplane if required. The modified algorithm is summarized as Algorithm 2 below.

Algorithm 2 (With cascaded pairs)**.** Initialization: set $\mathcal{M}^1=\mathcal{M}$, $\mathcal{Y}^1=\mathcal{Y}$, and $\mathcal{S}^1=\mathcal{S}$, define a list or a model of *correcting actions* – formalized alternations of the core AI in response in response to an error/error type.

Input to the ith iteration: Sets $\mathcal{M}^i$, $\mathcal{Y}^i$, and $\mathcal{S}^i=\mathcal{M}^i\cup\mathcal{Y}^i$; the number of clusters, p, and the value of filtering threshold, θ.

1–5. As in Algorithm 1.

6. *Training: Nodes creation and aggregation.* For each $\mathcal{Y}^i_{w,j}$, $j=1,\dots,p$ and its complement $\mathcal{S}^i_w\setminus\mathcal{Y}^i_{w,j}$ construct the following separating hyperplanes:

$$h_j(\boldsymbol{x})=\ell_j(\boldsymbol{x})-c_j,$$
$$\ell_j(\boldsymbol{x})=\left\langle\frac{\boldsymbol{w}_j}{\|\boldsymbol{w}_j\|},\boldsymbol{x}\right\rangle,\ c_j=\min_{\xi\in\mathcal{Y}^i_{w,j}}\left\langle\frac{\boldsymbol{w}_j}{\|\boldsymbol{w}_j\|},\xi\right\rangle$$
$$\boldsymbol{w}_j=(\mathrm{Cov}(\mathcal{S}^i_w\setminus\mathcal{Y}^i_{w,j})+\mathrm{Cov}(\mathcal{Y}^i_{w,j}))^{-1}(\bar{\boldsymbol{x}}(\mathcal{Y}^i_{w,j})-\bar{\boldsymbol{x}}(\mathcal{S}^i_w\setminus\mathcal{Y}^i_{w,j})).$$

Retain only those hyperplanes for which $c_j>\theta$. For each retained hyperplane, create a corresponding element $f_j(\cdot)$ in accordance to (16) or, if step 4 was used, then (17). For each retained hyperplane,

- determine the complementary set $\mathcal{C}_j$ comprised of elements $\boldsymbol{x}\in\mathcal{S}^i_w\setminus\mathcal{Y}^i_w$ for which $h_j(\boldsymbol{x})\geq 0$ (the set of points that are accidentally picked up by the f_j "by mistake")
- project elements of the set $\mathcal{C}_j\cup\mathcal{Y}^i_{w,j}$ orthogonally onto the hyperplane $h_j(\boldsymbol{x})=\ell_j(\boldsymbol{x})-c_j$ as

$$\boldsymbol{x}\mapsto\left(I-\frac{1}{\|\boldsymbol{w}_j\|^2}\boldsymbol{w}_j\boldsymbol{w}_j^T\right)\boldsymbol{x}+c_j\frac{\boldsymbol{w}_j}{\|\boldsymbol{w}_j\|}=P(\boldsymbol{w}_j)\boldsymbol{x}+b(\boldsymbol{w}_j,c_j);$$

- determine a hyperplane

$$h_{j,2}(\boldsymbol{x})=\ell_{j,2}(\boldsymbol{x})-c_{j,2},\ \ \ell_{j,2}(\boldsymbol{x})=\langle\boldsymbol{w}_{j,2},\boldsymbol{x}\rangle;$$

separating projections of $\mathcal{C}_j$ from that of $\mathcal{Y}^i_{w,j}$ so that $h_{j,2}(\boldsymbol{x})<0$ for all projections from $\mathcal{C}_j$ and $h_{j,2}(\boldsymbol{x})\geq 0$ for all projections from $\mathcal{Y}^i_{w,j}$. If no such planes exist, use linear Fisher discriminant or any other relevant computational procedure

- create a node

$$f_j^{\perp}(\boldsymbol{x})=f\left(\left\langle P(\boldsymbol{w}_j)WH^T(\boldsymbol{x}-\bar{\boldsymbol{x}}(\mathcal{S}^i))+b(\boldsymbol{w}_j,c_j),\boldsymbol{w}_{j,2}\right\rangle-c_{j,2}\right)$$

or, in case step 4 was used

$$f_j^{\perp}(\boldsymbol{x})=f\left(\left\langle P(\boldsymbol{w}_j)\varphi(WH^T(\boldsymbol{x}-\bar{\boldsymbol{x}}(\mathcal{S}^i)))+b(\boldsymbol{w}_j,c_j),\boldsymbol{w}_{j,2}\right\rangle-c_{j,2}\right);$$

Table 1
Error types in the system. Stars mark instances/events that are not accounted for.

Presence of a gesture	System's response	Error Type
Yes	Correctly classified	True Positive
	Incorrectly classified	False Positive
	Not reported	False Negative
No	Reported	False Positive*
	Not reported	True Negative*

- create the pair's response node, f_j^c, so that a non-negative response is generated only when both nodes produce a non-negative response

$$f_j^c(\boldsymbol{x}) = \quad f\big(\text{Step}(f_j(\boldsymbol{x})) + \text{Step}(f_j^{\perp}(\boldsymbol{x})) - 2\big);$$

- add the combined $f_j^c(\cdot)$ to the ensemble.

7. *Integration/Deployment.* Any $\boldsymbol{x}$ that is generated by the original core AI is put through the ensemble of $f_j^c(\cdot)$. If for some $\boldsymbol{x}$ any of the values of $f_j^c(\boldsymbol{x}) > 0$ then a *correcting action* is performed on the core AI. The action is dependent on the purpose of correction as well as on the problem at hand. It could include label swapping, signalling an alarm, ignoring/not reporting etc. The combined system becomes new core AI.
8. *Testing.* Assess performance of new core AI on a relevant data set. If needed, generate new sets $\mathcal{M}^{i+1}$, $\mathcal{Y}^{i+1}$, $\mathcal{S}^{i+1}$ (with possibly different error types and definitions), and repeat the procedure.

In the next section we illustrate how the proposed algorithms work in a case study example involving a core AI in the form of a reasonably large convolutional network trained on a moderate-size dataset.

4. Distinguishing the ten digits in american sign language: a case study

4.1. Setup and datasets

In this case study we investigated and tested the approach on a challenging problem of gestures recognition in the framework of distinguishing ten digits in American Sign Language. To apply the approach a core AI had to be generated first. As our core AI we picked a version of Inception deep neural network model [42] whose architecture is shown in Fig. 5. The model was trained[1] on ten sets of images that correspond to the American Sign Language pictures for 0–9 (see Fig. 6). Each set contained 1000 unique images consisting of profile shots of the persons hand, along with 3/4 profiles and shots from above and below.

The states $\boldsymbol{x}_i$ are the vectors containing the values of pre-softmax layer bottlenecks of size n for however many neurons are in the penultimate layer. Schematically the network's layer whose outputs are $\boldsymbol{x}_i$ is shown in the Diagram in Fig. 5 as the fully connected (bottleneck layer) in Unit H. Elements of this set that gave incorrect readings are noted and copied into the set $\mathcal{Y}$.

4.2. Experiments and results

Once the network was trained, additional 10,000 images of the same ratio were evaluated using the trained system. For these experiments the classification decision rule was to return a gesture number that corresponds to the network output with the highest score (winner-takes-all) if the highest score exceeds a given threshold, $\gamma \in [0, 1]$. Ties are broken arbitrarily at random. If the highest score is smaller than or equal to the threshold then no responses are returned.

To assess performance of the resulting classifier, we used the following indicators as functions of threshold γ:

$$\begin{aligned} \text{True Positive Rate } (\gamma) &= \frac{TP(\gamma)}{FN(\gamma) + TP(\gamma)}, \\ \text{Missclassification Rate } (\gamma) &= \frac{FP(\gamma)}{FP(0)}, \end{aligned} \tag{18}$$

where $TP\ (\gamma)$ is the number of true positives, $FN(\gamma)$ is the number of false negatives, and $FP(\gamma)$ is the number of false positives. The definitions of these variables are provided in Table 1.

In our experiments we did not add any negatives to the test set, as the original focus was to illustrate how the approach may cope with misclassification errors. This implies that the number of True Negatives is 0 for any threshold $\gamma \in [0, 1]$ and, consequently, the rate of false positives is always 1. Therefore, instead of using traditional ROC curves showing the rate of

[1] https://www.tensorflow.org/tutorials/image_retraining.

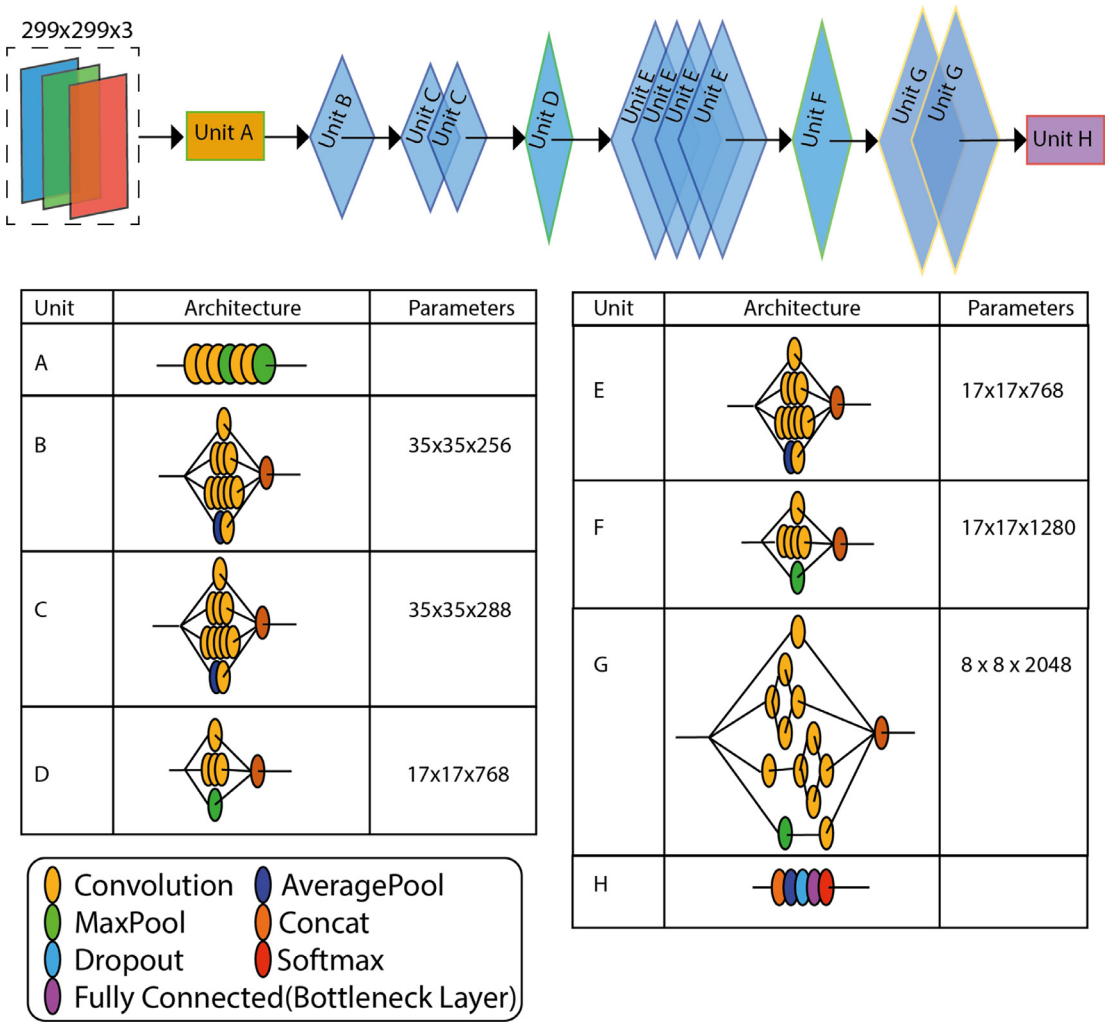

Fig. 5. Architecture of the adopted Inception deep neural network model. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

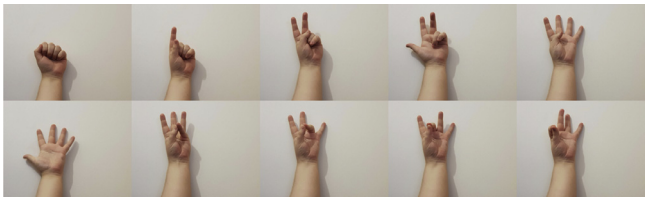

Fig. 6. Examples of the sort of images that appear in the current model's data set of the American Sign Language single hand positions for 0 (top left) to 9 (bottom right).

true positives against the rate of false positives, we employed a different family of curves in which the rate of false positives is replaced with the rate of misclassification as defined by (18).

A corresponding performance curve for this Core AI system is shown in Fig. 7. At $\gamma = 0$ the result was an 82.4% success rate for the adapted algorithm. At this end of the curve, the observed performance was comparable/similar to that reported in e.g. [7] (see also references therein). The numbers of errors per each gesture in the trained system are shown in Table 2. The variance of errors is mostly consistent among the ten classes with very few errors for the "0" gesture, likely due to its unique shape among the classes.

Once the errors were isolated, we used Algorithms 1 and 2 to construct correcting ensembles improving the original core AI. To train the ensembles, the testing data set of 10,000 images that has been used to assess performance of Inception was split into two non-overlapping subsets. The first subset, comprised of 6592 records of data points corresponding to

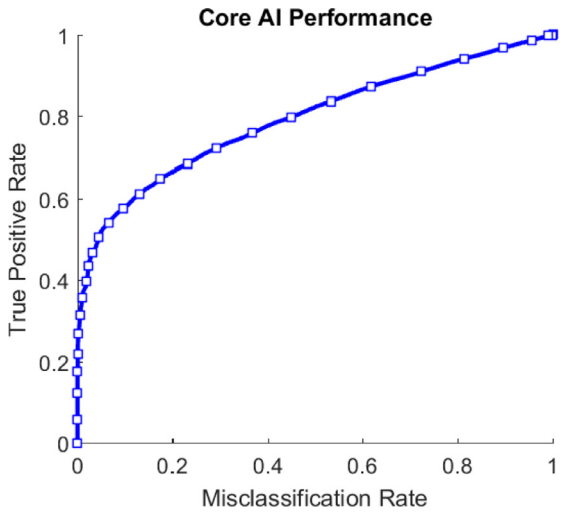

Fig. 7. Performance curve for the Core AI system (Inception) on the testing set of 10,000 images. Threshold γ was varying from 1 (bottom left corner) to 0 (top right corner).

Table 2
Errors per gesture (misclassifications). Top row corresponds to the gesture number, the bottom row indicates the number of errors for each gesture.

0	1	2	3	4	5	6	7	8	9
10	52	235	62	410	80	269	327	207	108

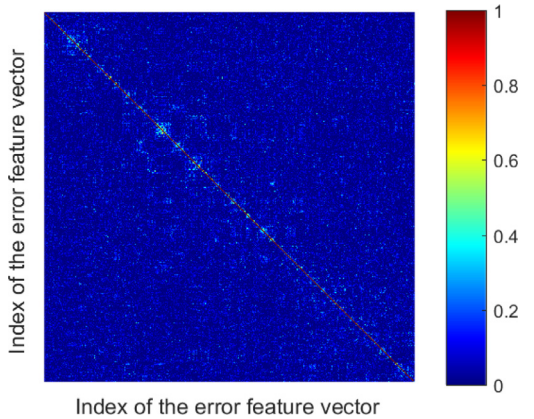

Fig. 8. Values of $|\langle \boldsymbol{x}_i/\|\boldsymbol{x}_i\|, \boldsymbol{x}_j/\|\boldsymbol{x}_j\|\rangle|$ (color-coded) for the data points labelled as errors. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

correct responses and 1408 records corresponding to errors, was used to train the correcting ensembles. This subset was the ensemble's training set, and it accounted for 80% of the data. The second subset, the ensemble's testing set, combining 1648 data points of correct responses and 352 elements labelled as errors was used to test the ensemble.

Both algorithms have been run on the first subset, the ensemble's training set. For simplicity of comparison and evaluation, we iterated the algorithms only once (i.e. did not build cascades of ensembles). In the regularization step, step 2, we used Kaiser–Guttman test. This returned 174 principal components reducing the original dimensionality more than 10 times. After the whitening transformation, step 3, we assessed the values of $|\langle \boldsymbol{x}_i/\|\boldsymbol{x}_i\|, \boldsymbol{x}_j/\|\boldsymbol{x}_j\|\rangle|$ (shown in Fig. 8). According to Fig. 8, data points labelled as errors are largely orthogonal to each other apart from few modestly-sized groups.

The number of clusters, parameter p in step 5, was varied from 2 to 1408 in regular increments. As a clustering algorithm we used standard k-means routine (k-means++) supplied with MATLAB 2016a. For each value of p, we run the k-means algorithm 10 times. For each clustering pass we constructed the corresponding nodes f_j (or their pairs for Algorithm 2) as prescribed in step 6, and combined them into a single correcting ensemble in accordance with step 7.

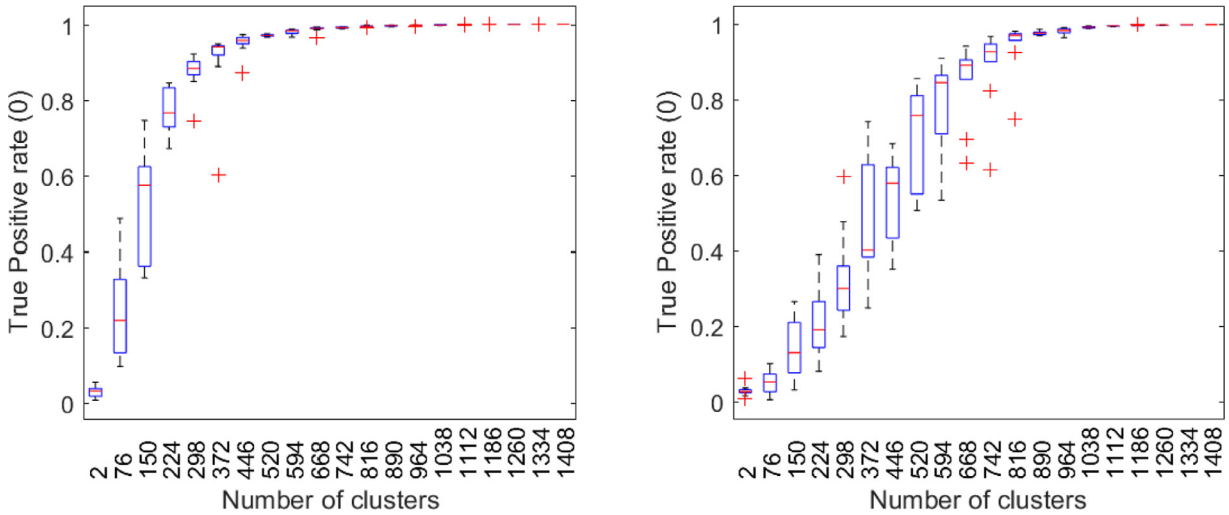

Fig. 9. True Positive rate at $\gamma = 0$ for the combined system as a function of the number of clusters, p. Left panel corresponds to Algorithm 1 *with* the optional projection step. Right panel corresponds to Algorithm 1 *without* the projection step.

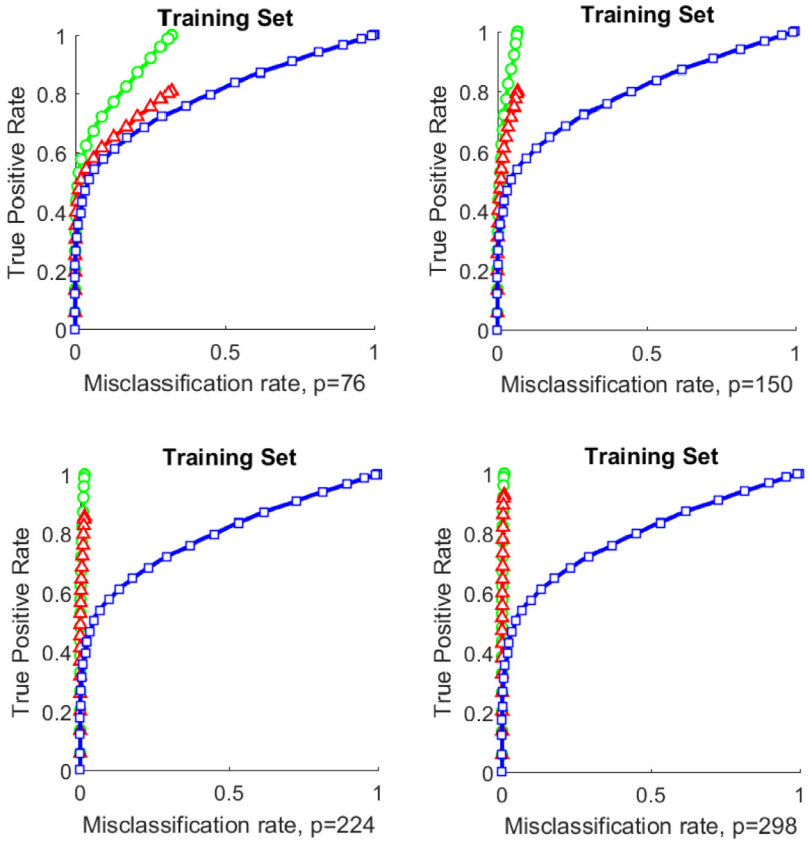

Fig. 10. Performance of the combined system after application of Algorithms 1 (red triangles) and 2 (green circles) on the training set for different numbers of clusters/nodes p in step 5. Blue squares show performance of core AI system. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

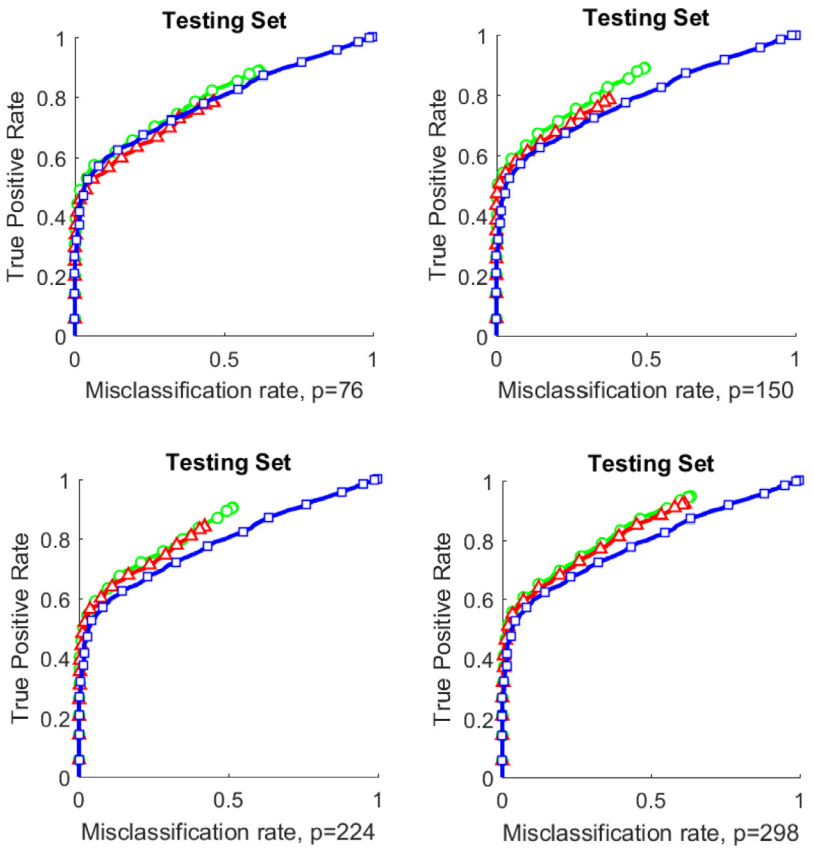

Fig. 11. Performance of the combined system after application of Algorithms 1 (red triangles) and 2 (green circles) on the testing set for different numbers of clusters/nodes p in step 5. Blue squares show performance of core AI system. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Before assessing performance of the ensemble, we evaluated filtering properties of the ensemble as a function of the number of clusters used. For consistency with predictions provided in Theorems 1–3, Algorithm 1 was used in this exercise. Results of the test are shown in Fig 9. Note that as the number p of clusters increases, the True Positive Rate at $\gamma = 0$ as a function of p, approaches 1 regardless of the projection step. This is in agreement with theoretical predictions stemming from Theorem 2. We also observed that performance drops rapidly with the average number of elements assigned to a cluster. In view of our earlier observation that vectors labelled as "errors" appear to be nearly orthogonal to each other, this drop is consistent with the bound provided in Corollary 1.

Next we assessed performance of Algorithms 1,2 and resulting ensembles on the training and testing sets for $p = 76, 150, 224, 298$. In both algorithms, optional projection step was used. The value of threshold θ was set to 0.2. In general, the value of threshold θ can be selected arbitrarily in an interval of feasible values of c_j. When the optional projection step is used, this interval is (0,1). Here we set the value of θ so that the hyperplanes $h_{j,2}$ in step 6 of Algorithm 2 produced by standard perceptron algorithm [39] consistently yielded perfect separation. Results are shown in Figs. 10 and 11. According to Fig. 10, performance on the training set grows with p, as expected. This is aligned with theoretical results presented in Section 3.1. The ensembles rapidly remove misclassification errors from the system, albeit at some cost to True Positive Rate. The latter cost for Algorithm 2 appears to be negligible. On the testing set, the picture changes. We see that, as the number p of clusters/nodes grows, performance of the ensemble saturates, with Algorithm 1 catching-up with Algorithm 2. This signals an over-fit and indicates a point where further increases in the number of nodes do not translate into expected improvements of the combined systems's performance. Apparent lack of correlations between feature vectors corresponding to errors (illustrated with Fig. 8) may also contribute to the observed performance saturation on the testing set.

Notably, for γ small both algorithms removed significant proportions of misclassification errors. This could be due to that training and testing sets for Algorithms 1 and 2 have been created from the performance analysis of the original core AI system at $\gamma = 0$.

5. Conclusion

In this work we presented a novel technology for computationally efficient improvements of generic AI systems, including sophisticated Multilayer and Deep Learning neural networks. These improvements are easy-to-train ensembles of elementary nodes. After the ensembles are constructed, a possible next step could be to further optimize the system via e.g. error back-propagation. Mathematical operations at the nodes involve computations of inner products followed by standard non-linear operations like ReLU, step functions, sigmoidal functions etc. The technology was illustrated with a simple case study confirming its viability. We note that the proposed ensembles can be employed for learning new tasks too.

The proposed concept builds on our previous work [22] and complements results for equidistributions in a unit ball to product measure distributions. When the clustering structure is fixed, the method is inherently one-shot and non-iterative, and its computational complexity scales at most linearly with the size of the training set. Sub-linear computational complexity of the ensembles construction makes the technology particularly suitable for large AI systems that have already been deployed and are in operation.

An intriguing and interesting feature of the approach is that training of the ensembles is largely achieved via Fisher linear discriminants. This, combined with the ideas from [23], paves the way for a potential relaxation of the i.i.d. assumption for the sampled data. Dealing with this as well as exploring the proposed technology on a range of different AI architectures and applications is the focus of our future work.

Acknowledgements

The work is supported by Innovate UK Technology Strategy Board (Knowledge Transfer Partnership grants KTP009890 and KTP010522) and by the Ministry of Education and Science of the Russian Federation (Project No. 14.Y26.31.0022). The authors are thankful to Tatiana Tyukina for help and assistance with preparing the manuscript and artwork.

Appendix A. Proofs of theorems and other technical statements

Proof of Theorem 1. The proof follows immediately from Theorem 2 in [20] and Hoeffding inequality [28]. Indeed, if $t>0$, X_i are independent bounded random variables, i.e. $a_i \le X_i \le b_i$, $i=1,\dots,n$, and $\overline{X}=1/n\sum_{i=1}^n X_i$ then (Hoeffding inequality)

$$P\big(\overline{X}-E[\overline{X}]\ge t\big)\le \exp\left(-\frac{2n^2t^2}{\sum_{i=1}^n (b_i-a_i)^2}\right)$$

$$P\big(|\overline{X}-E[\overline{X}]|\ge t\big)\le 2\exp\left(-\frac{2n^2t^2}{\sum_{i=1}^n (b_i-a_i)^2}\right).$$

Given that $x_{ij}=X_j$ where X_j are independent random variables with $-1\le X_j\le 1$, $E[X_j^2]=\sigma_j^2$ (Assumption 1), we observe that $\|\boldsymbol{x}_i\|^2=\sum_{j=1}^n X_j^2$

$$\begin{aligned} &P\left(\left|\frac{\sum_{j=1}^n X_j^2}{n}-E\left[\frac{\sum_{j=1}^n X_j^2}{n}\right]\right|\ge t\right)\\ &=P\left(\left|\frac{\sum_{j=1}^n X_j^2}{n}-\frac{R_0^2}{n}\right|\ge t\right)=P\left(\left|\frac{\|\boldsymbol{x}_i\|^2}{R_0^2}-1\right|\ge \frac{tn}{R_0^2}\right)\\ &\le 2\exp\left(-2nt^2\right).\end{aligned}$$

Denoting $\varepsilon=tn/R_0^2$ and recalling that $0\le X_j^2\le 1$ we conclude that (2) holds true.

Noticing that $E[x_{ik}x_{jk}]=E[x_{ik}]E[x_{jk}]=0$, $E[y_k x_{ik}]=0$, $-1\le x_{ik}\le 1$, $-1\le x_{jk}\le 1$, and $-1\le y_k x_{ik}\le 1$, we observe that estimates (3) and (4) follow. □

Proof of Theorem 2. Recall that for any events $A_1,\dots,A_k$ the following estimate holds:

$$P(A_1\&A_2\&\cdots\&A_k)\ge 1-\sum_{i=1}^k (1-P(A_i)). \tag{A.1}$$

Let

$$\frac{\|\boldsymbol{x}_{M+1}\|^2}{R_0^2}\ge 1-\varepsilon \quad (\text{event } A_1),$$

and

$$\left\langle \frac{\boldsymbol{x}_{M+1}}{R_0}, \frac{\boldsymbol{x}_i}{R_0}\right\rangle < 1-\varepsilon \quad (\text{events } A_2,\dots A_{M+1}).$$

The statement now follows from (2) and (3) with $\delta=1-\varepsilon$, and (A.1). □

Proof of Theorem 3. Suppose that $\|\boldsymbol{x}_i\|^2/R_0^2\ge 1-\varepsilon$ for all $\boldsymbol{x}_i\in\mathcal{Y}$ (event A_1). Then

$$\ell_k(\boldsymbol{x}_i) = \frac{1}{k}\left(\frac{\|\boldsymbol{x}_i\|^2}{R_0^2} + \sum_{\boldsymbol{x}_j\in\mathcal{Y},\ \boldsymbol{x}_j\neq\boldsymbol{x}_i}\left\langle\frac{\boldsymbol{x}_i}{R_0},\frac{\boldsymbol{x}_j}{R_0}\right\rangle\right) \geq \frac{1-\varepsilon+\beta(k-1)}{k}$$

for all $\boldsymbol{x}_i \in \mathcal{Y}$.

Note that $\bar{\boldsymbol{x}} \in [-1,1]^n$ by construction. Consider events

$$\ell_k(\boldsymbol{x}_j) = \left\langle\frac{\boldsymbol{x}_j}{R_0},\frac{\bar{\boldsymbol{x}}}{R_0}\right\rangle < \frac{1-\varepsilon+\beta(k-1)}{k} \text{ (events } A_{1+j})$$

$j = 1,\ldots,M$. According to Theorem 1 and (A.1)

$$P(A_1) \geq 1 - k\exp\left(-\frac{2R_0^4\varepsilon^2}{n}\right)$$

$$P(A_{1+j}) \geq 1 - \exp\left(-\frac{R_0^4(1-\varepsilon+\beta(k-1))^2}{2k^2n}\right).$$

Hence

$$P(A_1\&\cdots\&A_{1+M}) \geq$$

$$1 - k\exp\left(-\frac{2R_0^4\varepsilon^2}{n}\right) - M\exp\left(-\frac{R_0^4(1-\varepsilon+\beta(k-1))^2}{2k^2n}\right).$$

The result now follows. □

Proof of Corollary 1. Let $\|\boldsymbol{x}_i\|^2/R_0^2 \geq 1-\varepsilon$ for all $\boldsymbol{x}_i \in \mathcal{Y}$ (event A_1), and

$$\left\langle\frac{\boldsymbol{x}_i}{R_0},\frac{\boldsymbol{x}_j}{R_0}\right\rangle \geq -\delta \text{ for all } \boldsymbol{x}_i,\boldsymbol{x}_j\in\mathcal{Y},\ \boldsymbol{x}_i\neq\boldsymbol{x}_j \text{ (event } A_2).$$

The corollary follows immediately from (A.1) and Theorem 1 (Eqs. (2), (3)) □

Proof of Corollary 2. Let $\|\boldsymbol{x}_j\|^2 \geq 1-\varepsilon$. Then $h(\boldsymbol{x}) \geq 0$ for all $\boldsymbol{x}\in\Omega$. Estimate (13) hence follows from Theorem 1 and (A.1). □

Proof of Lemma 1. The proof follows that of Theorem 3 in [22]. Let $\boldsymbol{x}$ be an element of $\mathcal{M}$ and p_h be the probability of the event $h(\boldsymbol{x}) \geq 0$. Then the probability of the event that $h(\boldsymbol{x}) \geq 0$ for at most m elements from $\mathcal{M}$ is

$$P(M,m) = \sum_{k=0}^{m-1}\binom{M}{k}(1-p_h)^{M-k}p_h^k.$$

Observe that $P(M, m)$, as a function of p_h, is monotone and non-increasing on the interval [0,1], with $P(M,m) = 0$ at $p_h = 1$ and $P(M,n) = 1$ at $p_h = 0$. Hence

$$P(M,m) \geq \sum_{k=0}^{m-1}\binom{M}{k}(1-p_*)^{M-k}p_*^k = (1-p_*)^M\sum_{k=0}^{m-1}\binom{M}{k}\left(\frac{p_*}{1-p_*}\right)^k.$$

Given that $\frac{(M-m+1)^k}{k!} \leq \binom{M}{k} \leq \frac{M^k}{k!}$ for $0 \leq k \leq m-1$, we obtain

$$P(M,m) \geq (1-p_*)^M\sum_{k=0}^{m-1}\frac{1}{k!}\left(\frac{(M-m+1)p_*}{1-p_*}\right)^k.$$

Expanding e^x at $x = 0$ with the Lagrange remainder term:

$$e^x = \sum_{k=0}^{m-1}\frac{x^k}{k!} + \frac{x^m}{m!}e^\xi,\ \ \xi\in[0,x],$$

results in the estimate

$$\sum_{k=0}^{m-1}\frac{x^k}{k!} \geq e^x\left(1-\frac{x^m}{m!}\right) \text{ for all } x \geq 0.$$

Hence

$$P(M,m) \geq$$

$$(1-p_*)^M\exp\left(\frac{(M-m+1)p_*}{1-p_*}\right)\left(1-\frac{1}{m!}\left(\frac{(M-m+1)p_*}{(1-p_*)}\right)^m\right).$$

□

Appendix B. Hyperplanes vs ellipsoids for error isolation

Consider

$$C_n(\varepsilon) = B_n(1) \cap \left\{ \xi \in \mathbb{R}^n \middle| \left\langle \frac{\boldsymbol{x}}{\|\boldsymbol{x}\|}, \xi \right\rangle \geq 1 - \varepsilon \right\} \tag{B.1}$$

Let

$$\rho(\varepsilon) = (1 - (1-\varepsilon)^2)^{\frac{1}{2}}.$$

Note that $\rho(\varepsilon)$ is the radius of the ball containing the spherical cup C_n. Lemma 2 estimates volumes of spherical caps $C_n(\varepsilon)$ relative to relevant n-balls of radius $\rho(\varepsilon)$.

Lemma 2. *Let $C_n(\varepsilon)$ be a spherical cap defined as in (B.1), $\varepsilon \in (0, 1)$. Then*

$$\frac{\rho(\varepsilon)^{n+1}}{2}\left[\frac{1}{\pi^{\frac{1}{2}}}\frac{\Gamma\left(\frac{n}{2}+1\right)}{\Gamma\left(\frac{n}{2}+\frac{3}{2}\right)}\right] < \frac{\mathcal{V}(C_n(\varepsilon))}{\mathcal{V}(B_n(1))} \leq \frac{\rho(\varepsilon)^n}{2}.$$

Note that [4] $\mathcal{V}(B_n(r)) = r^n \mathcal{V}(B_n(1))$ for all $n \in \mathbb{N}$ $r > 0$. Hence the estimate of $\mathcal{V}(C_n(\varepsilon))$ from above is:

$$\mathcal{V}(C_n(\varepsilon)) \leq \frac{1}{2}\mathcal{V}(B_n(1))\rho(\varepsilon)^n. \tag{B.2}$$

Let us calculate the estimate of $\mathcal{V}(C_n(\varepsilon))$ from below. It is clear that

$$\mathcal{V}(C_n(\varepsilon)) = V(B_{n-1}(1)) \int_{1-\varepsilon}^{1} (1-x^2)^{\frac{n-1}{2}} dx$$

The integral in the right-hand side of the above expression can be estimated from below as

$$\int_{1-\varepsilon}^{1} (1-x^2)^{\frac{n-1}{2}} dx > \int_{1-\varepsilon}^{1} (1-x^2)^{\frac{n-1}{2}} x dx$$
$$= \frac{1}{2}\cdot\frac{1}{\frac{n}{2}+\frac{1}{2}}\cdot(1-(1-\varepsilon)^2)^{\frac{n+1}{2}} = \frac{1}{2}\cdot\frac{1}{\frac{n}{2}+\frac{1}{2}}\cdot\rho(\varepsilon)^{n+1}$$

Recall that $B_n(1) = \dfrac{\pi^{\frac{n}{2}}}{\Gamma\left(\frac{n}{2}+1\right)}$, $\Gamma(x+1) = x\Gamma(x)$. Hence

$$B_{n-1}(1)\cdot\frac{1}{\frac{n}{2}+\frac{1}{2}} = \frac{\pi^{\frac{n-1}{2}}}{\Gamma\left(\frac{n}{2}+\frac{1}{2}\right)}\frac{1}{\frac{n}{2}+\frac{1}{2}}$$
$$= \frac{\pi^{\frac{n-1}{2}}}{\Gamma\left(\frac{n+1}{2}+1\right)},$$

and

$$\int_{1-\varepsilon}^{1} (1-x^2)^{\frac{n-1}{2}} dx > \frac{1}{2}\mathcal{V}(B_n(1))\rho(\varepsilon)^{n+1}\left[\frac{1}{\pi^{\frac{1}{2}}}\frac{\Gamma\left(\frac{n}{2}+1\right)}{\Gamma\left(\frac{n}{2}+\frac{3}{2}\right)}\right]$$

□

Corollary 3. *Let $C_n(\varepsilon)$ be a spherical cap defined as in (B.1), $\varepsilon \in (0, 1)$, and $B_n(k\rho(\varepsilon))$ be an n-ball with radius $k\rho(\varepsilon)$, $k \in \mathbb{R}_{>0}$. Then*

$$\frac{\mathcal{V}(B_n(k\rho(\varepsilon)))}{\mathcal{V}(C_n(\varepsilon))} < k^n \frac{2\pi^{\frac{1}{2}}}{\rho(\varepsilon)}\left[\frac{\Gamma\left(\frac{n}{2}+1\right)}{\Gamma\left(\frac{n}{2}+\frac{3}{2}\right)}\right]^{-1}$$

Remark 3. Using Stirling's approximation we observe that

$$\frac{\Gamma\left(\frac{n}{2}+1\right)}{\Gamma\left(\frac{n}{2}+\frac{3}{2}\right)} = O\left(n^{-\frac{1}{2}}\right).$$

Thus $\frac{\mathcal{V}(B_n(\varepsilon))}{\mathcal{V}(C_n(\varepsilon))} < k^n H(n,\varepsilon)$ where $H(n,\varepsilon) = O\left(\frac{n^{1/2}}{\rho(\varepsilon)}\right)$.

According to Corollary 3 the volumes of $B_n(\varepsilon)$, $B_n(\kappa\rho(\varepsilon))$, $\kappa \in (0, 1)$ decay exponentially with dimension n relative to that of $C_n(\varepsilon)$. This implies that distance-based detectors are extremely localized, and in comparison with simple perceptrons, the proportion of points to which they respond positively is negligibly small. On the other hand, filtering properties of hyperplanes are extreme in high dimension (see Theorems 2, 3 in Section 3.1). This combination of properties makes perceptrons and their ensembles particularly attractive for fine-tuning of existing AI systems.

References

[1] J. Anderson, M. Belkin, N. Goyal, L. Rademacher, J. Voss, The more, the merrier: the blessing of dimensionality for learning large Gaussian mixtures, J. Mach. Learn. Res. 35 (2014) 1–30.
[2] D. Arthur, S. Vassilvitskii, k-means++: The advantages of careful seeding, in: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035.
[3] D. Arthur, S. Vassilvitskii, How slow is the k-means method? in: Proceedings of the twenty-second annual symposium on Computational geometry, ACM, 2006, pp. 144–153.
[4] K. Ball, An elementary introduction to modern convex geometry, Flavors of Geom. 31 (1997) 1–58.
[5] A.R. Barron, Universal approximation bounds for superposition of a sigmoidal function, IEEE Trans. Inf. Theory 39 (3) (1993) 930–945.
[6] R. Beene, A. Levin, E. Newcomer, Uber self-driving test car in crash wasn't programmed to brake, 2018, (https://www.bloomberg.com/news/articles/2018-05-24/uber-self-driving-system-saw-pedestrian-killed-but-didn-t-stop).
[7] V. Bheda, D. Radpour, Using deep convolutional networks for gesture recognition in American Sign Language, 2017, arXiv preprint arXiv:1710.06836.
[8] P.P. Brahma, D. Wu, Y. She, Why deep learning works: a manifold disentanglement perspective, IEEE Trans. Neural Netw. Learn. Syst. 27 (10) (2016) 1997–2008.
[9] O. Chapelle, Training a support vector machine in the primal, Neural Comput. 19 (5) (2007) 1155–1178.
[10] T. Chen, I. Goodfellow, J. Shlens, Net2net: accelerating learning via knowledge transfer, ICLR 2016 (2015) arXiv preprint arXiv:1511.05641.
[11] F. Cucker, S. Smale, On the mathematical foundations of learning, Bull. Am. Math. Soc. 39 (1) (2002) 1–49.
[12] D. Donoho, High-dimensional data analysis: the curses and blessings of dimensionality, AMS Math Challenges Lect. 1 (2000) (2000) 32.
[13] T.J. Draelos, N.E. Miner, C.C. Lamb, C.M. Vineyard, K.D. Carlson, C.D. James, J.B. Aimone, Neurogenesis deep learning, IEEE International Joint Conference on Neural Networks (IJCNN), IEEE, 2017, pp. 526–533.
[14] S.E. Fahlman, C. Lebiere, The cascade-correlation learning architecture, in: Advances in neural information processing systems, 1990, pp. 524–532.
[15] C. Foxx, Face recognition police tools "staggeringly inaccurate", 2018, (http://www.bbc.co.uk/news/technology-44089161).
[16] Y. Freund, R.E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, J. Comput. Syst. Sci. 55 (1) (1997) 119–139.
[17] J. Gama, P. Brazdil, Cascade generalization, Mach. Learn. 41 (3) (2000) 315–343.
[18] J. Gibbs, Elementary Principles in Statistical Mechanics, Developed with Especial Reference to the Rational Foundation of Thermodynamics, Dover Publications, New York, 1960 (1902).
[19] A.N. Gorban, Order-disorder separation: geometric revision, Phys. A 374 (2007) 85–102.
[20] A.N. Gorban, I.Y. Tyukin, Stochastic separation theorems, Neural Netw. 94 (2017) 255–259.
[21] A.N. Gorban, I.Y. Tyukin, Blessing of dimensionality: mathematical foundations of the statistical physics of data, Philos. Trans. R. Soc. A 376 (2018).
[22] A.N. Gorban, R. Burton, I. Romanenko, I.Y. Tyukin, One-trial correction of legacy AI systems and stochastic separation theorems, Inf. Sci. 484 (2019) 237–254.
[23] A.N. Gorban, A. Golubkov, B. Grechuk, E.M. Mirkes, I.Y. Tyukin, Correction of AI systems by linear discriminants: probabilistic foundations, Inf. Sci. 466 (2018) 303–322.
[24] M. Gromov, Metric Structures for Riemannian and non-Riemannian Spaces. With appendices by M. Katz, P. Pansu, S. Semmes. Translated from the French by Sean Muchael Bates, Birkhauser, Boston, MA, 1999.
[25] M. Gromov, Isoperimetry of waists and concentration of maps, GAFA, Geom. Funct. Anal. 13 (2003) 178–215.
[26] J.A. Hartigan, M.A. Wong, A k-means clustering algorithm, J. R. Stat. Soc. Ser. C (Applied Statistics) 28 (1) (1979) 100–108.
[27] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778.
[28] W. Hoeffding, Probability inequalities for sums of bounded random variables, J. Am. Stat. Assoc. 58 (301) (1963) 13–30.
[29] F.N. Iandola, S. Han, M.W. Moskewicz, K. Ashraf, W.J. Dally, K. Keutzer, SqueezeNet: Alexnet-level accuracy with 50x fewer parameters and $<$ 0.5 mb model size, 2016, arXiv preprint arXiv:1602.07360.
[30] D. Jackson, Stopping rules in principal components analysis: acomparison of heuristical and statistical approaches, Ecology 74 (8) (1993) 2204–2214.
[31] P.C. Kainen, Utilizing geometric anomalies of high dimension: when complexity makes computation easier, in: Computer Intensive Methods in Control and Signal Processing, Springer, 1997, pp. 283–294.
[32] P.C. Kainen, V. Kurkova, Quasiorthogonal dimension of euclidian spaces, Appl. Math. Lett. 6 (3) (1993) 7–10.
[33] A. Kuznetsova, S. Hwang, B. Rosenhahn, L. Sigal, Expanding object detectors horizon: incremental learning framework for object detection in videos, in: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 28–36.
[34] P. Lévy, Problèmes concrets d'analyse fonctionnelle, 2nd, Gauthier-Villars, Paris, 1951.
[35] I. Misra, A. Shrivastava, M. Hebert, Semi-supervised learning for object detectors from video, in: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 3594–3602.
[36] L. Pratt, Discriminability-based transfer between neural networks, Adv. Neural Inf. Process. (5) (1992) 204–211.
[37] A. Prest, C. Leistner, J. Civera, C. Schmid, V. Ferrari, Learning object class detectors from weakly annotated video, in: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2012, pp. 3282–3289.
[38] L. Rokach, Ensemble-based classifiers, Artif. Intell. Rev. 33 (1–2) (2010) 1–39.
[39] F. Rosenblatt, Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms, Spartan Books, 1962.
[40] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A.C. Berg, L. Fei-Fei, Imagenet large scale visual recognition challenge, Int. J. Comput. Vis. (2014) 1–42, doi:10.1007/s11263-015-0816-y.
[41] S. Scardapane, D. Wang, Randomness in neural networks: an overview, Data Min. Knowl. Discov. 7 (2) (2017).
[42] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 1–9.
[43] V. Vapnik, O. Chapelle, Bounds on error expectation for support vector machines, Neural Comput. 12 (9) (2000) 2013–2036.
[44] V. Vapnik, R. Izmailov, Knowledge transfer in SVM and neural networks, Ann. Math. Artif. Intell. (2017) 1–17.
[45] D. Wang, C. Cui, Stochastic configuration networks ensemble with heterogeneous features for large-scale data analytics, Inf. Sci. 417 (2017) 55–71.
[46] D. Wang, M. Li, Deep stochastic configuration networks with universal approximation property, 2017. (An updated version of this work has been published in the Proceedings of 2018 IJCNN). arXiv: 1702.05639.
[47] D. Wang, M. Li, Stochastic configuration networks: fundamentals and algorithms, IEEE Trans. Cybern. 47 (10) (2017) 3466–3479.
[48] J. Yosinski, J. Clune, Y. Bengio, H. Lipson, How transferable are features in deep neural networks? in: Advances in neural information processing systems, 2014, pp. 3320–3328.
[49] S. Zheng, Y. Song, T. Leung, I. Goodfellow, Improving the robustness of deep neural networks via stability training, in: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 4480–4488. https://arxiv.org/abs/1604.04326.