Flexible Kernel Memory

Dimitri Nowicki[1,2]*, **Hava Siegelmann**[1,3]

1 Biologically Inspired Neural and Dynamical Systems (BINDS) Lab, Department of Computer Science, University of Massachusetts Amherst, Amherst, Massachusetts, United States of America, **2** Institute of Mathematical Machines and Systems Problems of Ukraine National Academy of Science (IMMSP NASU), Center for Cybernetics, Kiev, Ukraine, **3** Program on Evolutionary Dynamics, Harvard University, Cambridge, Massachusetts, United States of America

Abstract

This paper introduces a new model of associative memory, capable of both binary and continuous-valued inputs. Based on kernel theory, the memory model is on one hand a generalization of Radial Basis Function networks and, on the other, is in feature space, analogous to a Hopfield network. Attractors can be added, deleted, and updated on-line simply, without harming existing memories, and the number of attractors is independent of input dimension. Input vectors do not have to adhere to a fixed or bounded dimensionality; they can increase and decrease it without relearning previous memories. A memory consolidation process enables the network to generalize concepts and form clusters of input data, which outperforms many unsupervised clustering techniques; this process is demonstrated on handwritten digits from MNIST. Another process, reminiscent of memory reconsolidation is introduced, in which existing memories are refreshed and tuned with new inputs; this process is demonstrated on series of morphed faces.

Citation: Nowicki D, Siegelmann H (2010) Flexible Kernel Memory. PLoS ONE 5(6): e10955. doi:10.1371/journal.pone.0010955

Editor: Pedro Antonio Valdes-Sosa, Cuban Neuroscience Center, Cuba

Received September 7, 2009; **Accepted** March 23, 2010; **Published** June 11, 2010

Funding: This work was supported by the Office of Naval Research grant #109-0138R (www.onr.navy.mil). The funder had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing Interests: The authors have declared that no competing interests exist.

* E-mail: nowicki@cs.umass.edu

Introduction

Memory experiments demonstrated persistent activity in several structures in the lower mammal, primate, and human brains including the hippocampus [1,2], prefrontal [3], visual [4] and oculomotor cortex [5], basal ganglia [6], etc. Persistent dynamics is believed to emerge as attractor dynamics (see also [7]–[13]). Currently, the leading paradigm in attractor neural networks memory models is the Hopfield model [14] with possible variations, including activation functions, neural firing, density of the neural connections, and the memory loading paradigms [15]–[22].

In this paper, we introduce a memory model, in which its memory attractors do not lie in the input or neural space as in classical models but rather in a feature space with large or infinite dimension. This model is isomorphic to a symmetric Hopfield network in the kernels' φ-space, giving rise to a Lyapunov function in the dynamics of associative recalls, which enables the analogy to be drawn between memories and attractors in the kernel space.

There are several advantages to our novel kernel approach to attractor memory. The input space can be composed of either continuous-valued or binary vectors. The number of attractors m is independent of the input dimension n, thus posing a saturated-free model, which does not suffer corrupted memories with memory overload. Attractors can be efficiently loaded, deleted, and updated on-line, something that has previously been only a property of symbolic computer-memory models. Furthermore, for the first time in neural memory models, we have demonstrated a method allowing input dimension not to be constrained to a fixed size or be a priori bounded; dimension can change with time, similar to organic memory allocation for memories of greater importance or increased detail. These attributes may be very beneficial in psychological modeling.

The process of kernel memory consolidation results in attractors in feature space and Voronoi-like diagrams that can be projected efficiently to the input space. The process can also describe clusters, which enables the separation of even very close-by attractors. Another re-consolidation process enables tracking monotonic updates in inputs including moving and changing objects.

Generalizing Radial-Basis-Function Networks

Our network can be thought of as generalizing Radial Basis Function (RBF) networks [23]. These are 2-layered feed-forward networks with the first layer of neurons having linear activation and the second layer consisting of neurons with RBF activation function. Recurrent versions of the RBF networks [24,25] add time-delayed feedback from the second to the first layer. Our network enables a more generalized structure, both in terms of number of layers and in allowing for many more general activation functions.

Unlike previous RBF networks, our activation functions are chosen from a large variety of kernels that allow us to distinguish attractors that are similar or highly correlated. Furthermore, unlike any previous RBF network with its fixed architecture and activation functions, our selected neural kernel functions can change during learning to reflect the memory model's changing properties, dimension, or focus. We go on to prove that the attractors are either fixed points or 2-cycles, unlike general recurrent RBF networks that may have arbitrary chaotic attractors [26,27]; regular attractors are advantageous for a memory system.

Synaptic plasticity and Memory Reconsolidation

Reconsolidation is a process occurring when memory becomes liable during retrieval and can then be updated. This process is implicated in learning and flexible memories when healthy; it leads to amnesia and compulsive disorders when corrupted. Reconsolidation is observed both in neurophysiological and psychological studies ([28]–[33]) and has been modeled in artificial neural systems as well ([19,34]). While the actual processes underlying reconsolidation are still being studied, the property of dependance on sample ordering has been established in both electrophysiology of CA3 neurons [13] and in psychophysics [35]. In reconsolidation, memory representations are sensitive to the order of sample data: when samples change in an orderly manner, the reconsolidation process learns and updates effectively. When samples are shuffled and consistent direction of change is lost, existing memories do not update. We show here that the importance of input ordering is inherent in any update processes, reminiscent of reconsolidation. We also demonstrate how reconsolidation works in flexible environments and with large-scale data beyond the model shown in [19].

Our flexible model assumes global memory update. This is an interesting approach for a few reasons. First, it results in more stable and robust updates: in other models the "closest" attractor may be selected incorrectlyly due to noise. Second, it enables a direct analogy to an existing neural model of reconsolidation [19] since there the whole synaptic matrix is adjusted, not simply a chosen attractor. Moreover with global updates our memory can demonstrate phenomena analogous to the gang effect [36]. While we have taken a global update approach our model retains the property in which the retrieved attractor (the attractor closest to the current input) is most affected.

Kernel Based Algorithms and Memories

The memory system introduced here takes advantage of developments introduced in Support Vector Machine (SVM) [37], Least-Square SVM [38] and Support Vector Clustering [39], where kernel functions enable data handling in higher feature spaces for richer boundaries, yet do so efficiently and cheaply. Our support-vector-like memory system incorporates the realistic property of flexible attractors with high dimensional feature spaces, while being tractable and implementable by neural units.

Zhang et al. [40] introduced a feedforward network with particular kernels in the form of wavelet and Radial Basis Functions (RBF) that were fit to perform a face recognition task efficiently. The kernel heteroassociative memories were organized into the modular network. Our architecture can be recurrent, which is more powerful than the feedforward method, can handle discrete and analog inputs, and the kernels we use can change online adding increased flexibility and capability.

Caputo [41] explored analogies from associative memory to "kernel spin glass" and demonstrated an attractor network, loading bipolar inputs and using generalized Hebbian learning to load non-flexible memories with greater memory capacity than the Hopfield network. In this work, a kernel algorithm generalized the Hebbian rule and the energy function of Hopfield networks, while capacity estimations generalized Amit's approach [1]. This method built in the free energy function in addition to the Hamiltonian. Our system, by comparison, allows for both binary and continuous inputs, is far more flexible in that the kernels adapt themselves over time and that attractors and features can be added and removed. Further, our system is more practical in that it has the added capability to cluster data.

Support vector memory by Casali et al. [42] utilized support vectors to find the optimal symmetric Hopfield-like matrix for a set of binary input vectors. Their approach is very different from ours despite the similar title, in that it considers only binary symmetric case and has bounded attraction space. Support-vector optimization is used to find optimal matrix $\mathbf{W}$ for given $m \times n$ matrix $\mathbf{X}$ of etalons. This matrix must satisfy relationship $\mathbf{sign}(\mathbf{WX}) = \mathbf{sign}(\mathbf{X})$. Support vectors are found to provide optimal margins of $\mathbf{WX}$. Kernels are not used in this work and hence the name is somewhat confusing. Our kernel memory is far richer: the number of memory attractors is not bounded by input dimension - accomplished by varying the input space; our encoding is more efficient, our memory can use discrete or analog space, one-shot learning, and overall is more flexible.

In support vector clustering [39], clusters are formed when a sphere in the φ-space spanned by the kernels is projected efficiently to input space. Here the clustering is a side effect of the created memories that are formed as separated fixed points in the φ-space, and where the Voronoi polyhedron is projected on a formation of clusters in the input space. Formation of memories is local, sharing this concept with the Localist Attractor Network [43] and the Reconsolidation Attractor Network [34].

Organization

This work will be presented as follows: At first, the model of kernel heteroassociative memory is introduced, followed by the special case of auto-associative memory where attractors emerge. A neural representation is layered, and robustness (attraction radius) is estimated. We then introduce a technique that allows adding and removing attractors to the existing kernel associative network, and follow by introducing another technique that adds or removes input dimensionality on line. We next show a procedure of consolidating data into representing attractors, and demonstrate clusters emerging on handwritten digits and conclude by introducing the functional level of reconsolidation in memory and applications to morphed faces.

Results and Discussion

2.1 Our Kernel Heteroassociative Memory Model

A general framework of heteroassociative memory can be defined on the input E_x and output E_y spaces, with dimensionalities n and p respectively, and with m pairs of vectors $\mathbf{x}_i \in E_x$, $\mathbf{y}_i \in E_y$, $i = 1 \dots m$ to be stored. The input vectors in the space E_x can be written as columns of matrix $\mathbf{X}$ $(n \times m)$ and associated vectors in the output space E_y as columns of matrix $\mathbf{Y}$ $(p \times \mathrm{m})$. A projective operator $\tilde{\mathbf{B}}: E_X \to E_Y$ such that $\tilde{\mathbf{B}}\mathbf{x}_i = \mathbf{y}_i$ can be written in a matrix form $\tilde{\mathbf{B}}$ with

$$\tilde{\mathbf{B}}\mathbf{X} = \mathbf{Y} \tag{1}$$

and be solved as

$$\mathbf{B} = \mathbf{Y}\mathbf{X}^{+} \tag{2}$$

with "$^{+}$" stands for the Moore-Penrose pseudoinverse of $\mathbf{X}$ [44]. If the columns of X are linearly independent, the pseudoinverse matrix can be calculated by

$$\mathbf{X}^{+} = (\mathbf{X}^{T}\mathbf{X})^{-1}\mathbf{X}^{T}. \tag{3}$$

Let us define matrix $\tilde{\mathbf{S}}$, $(m \times m)$, where the elements $\tilde{s}_{ij}$ are the pairwise scalar products of the memorized vectors, that is

$\tilde{s}_{ij} = (\mathbf{x}_i, \mathbf{x}_j)$, or in matrix notation:

$$\tilde{\mathbf{S}} = \mathbf{X}^T \mathbf{X}.$$

Then $\tilde{\mathbf{B}}$ can be written as:

$$\tilde{\mathbf{B}} = \mathbf{Y}\tilde{\mathbf{S}}^{-1}\mathbf{X}^T. \tag{4}$$

We propose to formulate the pseudoinverse memory association (recall) by calculating for each input vector $\mathbf{x} \in E_x$ the output by:

$$\begin{aligned} &\mathbf{y} = \tilde{\mathbf{B}}\mathbf{x} = \mathbf{Y}\tilde{\mathbf{S}}^{-1}\mathbf{z}; \\ &\mathbf{z} = \mathbf{X}^T\mathbf{x}; \\ &z_i = (\mathbf{x}_i, \mathbf{x}). \end{aligned} \tag{5}$$

This is a "one-pass" non-iterative linear associative memory. It has the property that if two input samples are close to each other, then the two outputs will be close to each other as well: $\|\mathbf{y}' - \mathbf{y}\| \leq \|\mathbf{B}\| \cdot \|\mathbf{x}' - \mathbf{x}\|$.

2.1.1 Memory in Feature Space. In order to overcome the common dependence of memory capacity on input dimension, we transform the input space E_X to a new input space $E\varphi$ which we call feature space, whose dimensionality can be far greater than the dimension of E_X, n (it could even be an infinite-dimensional Hilbert space). The transformation φ: $E_X \rightarrow E\varphi$ is considered to be transferring from input to feature space.

The respective associative memory algorithm can now be defined as follows:

$$\mathbf{B}\varphi(\mathbf{X}) = \mathbf{Y} \tag{6}$$

$$\mathbf{B} = \mathbf{Y}[\varphi(\mathbf{X})]^+ \tag{7}$$

$$[\varphi(\mathbf{X})]^+ = ([\varphi(\mathbf{X})]^T\varphi(\mathbf{X}))^{-1}[\varphi(\mathbf{X})]^T \tag{8}$$

Analogously writing $\mathbf{S}$ as

$$\begin{aligned} &\mathbf{S} = [\varphi(\mathbf{X})]^T\varphi(\mathbf{X}) \\ &s_{ij} = (\varphi(\mathbf{x}_i), \varphi(\mathbf{x}_j)), \end{aligned} \tag{9}$$

the memory loads by:

$$\mathbf{B} = \mathbf{Y}\mathbf{S}^{-1}[\varphi(\mathbf{X})]^T. \tag{10}$$

and the association (recall) procedure is calculated by:

$$\begin{aligned} &\mathbf{y} = \mathbf{B}\mathbf{x} = \mathbf{Y}\mathbf{S}^{-1}\mathbf{z}; \\ &\mathbf{z} = [\varphi(\mathbf{X})]^T\varphi(\mathbf{x}); \\ &z_i = (\varphi(\mathbf{x}_i), \varphi(\mathbf{x})). \end{aligned} \tag{11}$$

Remark 1 Linear independence of the vectors $\varphi(\mathbf{x}_i)$ in the φ-space is required in order to use identity (3) for the Moore-Penrose pseudoinverse (see [44]). It is achieved as we will see below by using piece-wise Mercer kernels, and does not limit the number of attractors. This identity is used to bring equation (7) to the form of (8) and to introduce $\mathbf{S}$.

We note that during both loading (10) and recall (11) procedures, the function φ appears in the pair $(\varphi(\mathbf{x}_i), \varphi(\mathbf{x}_j))$. We can thus define a Kernel function over $E_x \times E_x$ and gain computational advantage.

Let us denote a scalar product in the *feature space* $E\varphi$ by $K(\mathbf{u}, \mathbf{v}) = (\varphi(\mathbf{u}), \varphi(\mathbf{v}))$. This is a symmetric, real-valued, and nonnegative-definite function over $E_x \times E_x$ called a *kernel* [37]. We now can write $\mathbf{S}$ and $\mathbf{z}$ using the Kernel K:

$$\begin{aligned} &s_{ij} = K(\mathbf{x}_i, \mathbf{x}_j);\, i,j = 1 \ldots m \\ &z_i = K(\mathbf{x}_i, \mathbf{x}). \end{aligned} \tag{12}$$

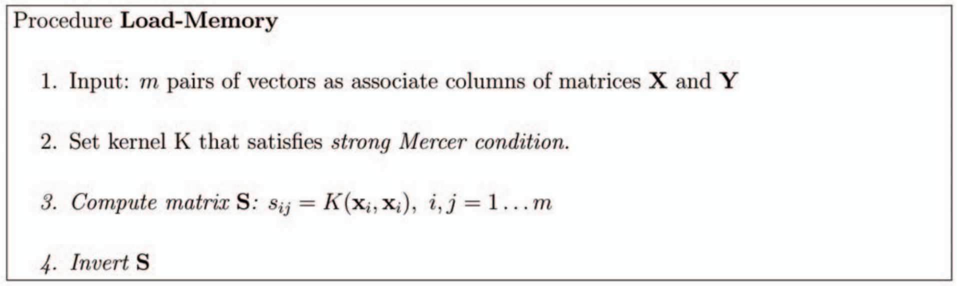
The value of the Kernel function is scalar. Thus even if φ was a function of high dimension the calculation of the multiplication is a scalar and thus fast to calculate.

Mercer kernels as used in Support Vector Machines [37] are not sufficient for creating the associative memory we introduce, since our memories also require that all attractors are linearly independent in the feature space. To enable such independence we define the piece-wise Mercer kernels and in Section "Piece-wise Mercer Kernels" of Materials and Methods (MM) we prove that they can always be found and always lead to independence.

As opposed to Hebbian learning that requires $O(n^2 m)$ multiplications, we need $O(m^3 + nm^2)$ arithmetic operations over real scalars. The loading algorithm is displayed in Fig. 1. The memory is proven below to associate loaded pairs correctly and to associate close by values otherwise (see Materials and Methods (MM)).

2.1.2 Memory Independent on Input Dimension. The kernel heteroassociative memory has no a priori bound on capacity in the following sense: for any given learning sample there exists a kernel such that the memory with this kernel will provide the desired association.

To specify this we formulate the following theorem, which is proven in MM Section "Correctness of Association":



Procedure **Load-Memory**

1. Input: m pairs of vectors as associate columns of matrices $\mathbf{X}$ and $\mathbf{Y}$
2. Set kernel K that satisfies *strong Mercer condition.*
3. *Compute matrix* $\mathbf{S}$*:* $s_{ij} = K(\mathbf{x}_i, \mathbf{x}_i),\ i, j = 1 \ldots m$
4. *Invert* $\mathbf{S}$

Figure 1. The algorithm of memory loading.
doi:10.1371/journal.pone.0010955.g001

Theorem 1 *For any memory size m, let $(\mathbf{x}_i,\mathbf{y}_i)\in E_x\times E_y$, $i=1\ldots m$ be a learning sample consisting of m input-output pairs. Then there exists a piece-wise Mercer kernel K such that the associative memory that has this kernel and governed by equations (9)–(11) assigns $\mathbf{x}_i$ to $\mathbf{y}_i$ for all $i=1\ldots m$.*

Remark 2 For the correct association, the memories have to be linearly independent in the feature space. As we have shown here this does not pose a memory limit, because for any given learning sample we can find a (piece-wise Mercer) kernel that guarantees such independence.

2.2 The Kernel Autoassociative Memory Model

We next focus on the special case where $E_x=E_y$, and the stored vectors $\mathbf{x}_\mu=\mathbf{y}_\mu\equiv\xi^\mu$, $\mu=1\ldots m$. Here the loading algorithm is the same as in Fig. 1, and recall is facilitated by the iterative form:

$$\mathbf{x}_{t+1}=f(\mathbf{y}_t). \qquad (13)$$

The activation function $f(x)$ is applied by coordinates and constitutes a bounded monotonically increasing real-valued function over $\mathbb{R}$ such that $\lim_{x\to-\infty}f(x)=a$, and $\lim_{x\to+\infty}f(x)=b$, $b>a$.

The scheme of kernel auto-associative memory working in recall mode is shown in Fig. 2. We prove (in lemma 4 in MM Section "Proving Convergence of the Autoassociative Recall Algorithm") that the recall procedure always converges and that the attractors are either fixed points or 2-limit cycles. Joining all operations to a single equation we get:

$$\mathbf{x}(t+1)=f\left(\sum_{\mu,\nu=1}^{m}\xi^\mu\hat{s}_{\mu,\nu}K(\xi^\nu,\mathbf{x})\right) \qquad (14)$$

Here by $\hat{s}_{\mu,\nu}$ we denote the elements of $\mathbf{S}^{-1}$. In coordinate form this equation is:

$$x_i(t+1)=f\left(\sum_{\mu,\nu=1}^{m}\xi_i^\mu\hat{s}_{\mu,\nu}K(\xi^\nu,\mathbf{x})\right) \qquad (15)$$

The pseudocode of the associative recall is shown in the Fig. 3. As will be shown in the next section, the double nonlinearity of the recall dynamics does not reduce the biological plausibility since the kernel memory can be designed as a layered neural network with only one nonlinear operation per neuron.

In Materials and Methods, Section "Example of the associative recall", we provide an explicit example of kernel autoassociative memory with $E_x=\mathbb{R}^3$ and $E\varphi=\mathbb{R}^6$. We demonstrate there how a set of five vectors is memorized and how the iterative recall works.

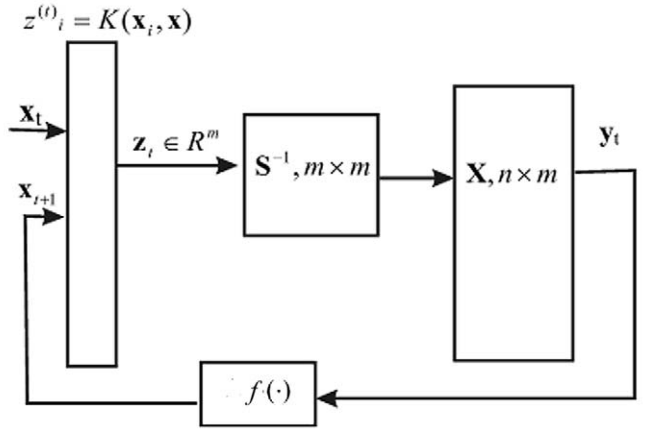


Figure 2. Scheme of the kernel autoassociative memory.
doi:10.1371/journal.pone.0010955.g002

2.3 Kernel Associative Memory as a Neural Network

The autoassociative kernel memory can be directly implemented in a recurrent layered neural network (Fig. 4a): The network has n inputs. The first layer has m neurons that perform kernel calculations; the i-th neuron computes $z_i=K(\mathbf{x},\mathbf{x}_i)$. In the special case where the kernel is a radial-basis function $K(\mathbf{u},\mathbf{v})=R(\|\mathbf{u}-\mathbf{v}\|)$ these neurons are the common RBF neurons [23]. The second layer has m neurons, its weight matrix is $\mathbf{S}^{-1}$. The neurons of the second layer can be either linear or have a generalized sigmoid activation function.

The third layer also has n neurons, its weight matrix is $\mathbf{X}^T$. Its activation function can be linear, generalized sigmoid or the even more general sigmoid from Equation (15) above. The network has "one-to-one" feedback connections from the last layer to the inputs. In recall mode it works in discrete time, like Hopfield networks.

Definition 1 *A monotonic bounded piecewise-differentiable function $f:\mathbb{R}\to\mathbb{R}$ such that $f(0)=0$, $f(1)=1$, and $f'<\theta<1$ in certain neighborhoods of 0 and 1 is called generalized sigmoid.*

Theorem 2 *Suppose that the kernel associative memory has a generalized sigmoid activation function in the second layer. Then the attractors emerging by the iterative recall procedure are either fixed points or 2-cycles.*

Proof appears in Material and Methods Section "Proving Convergence of the Autoassociative Recall Algorithm".

2.3.1 The Attraction Radius. A key question for any neural network or learning machine is how robust it is in the presence of noise. In attractor networks, the stability of the associative retrieval and the robustness to noise can be measured by the *attraction radius*.

Definition 2 *Suppose the input to an attractor network belongs to the metric space with distance ρ. For an attractor ξ_μ of the network let R_μ be the largest positive real number such that if $\rho(\xi_\mu,x)<R_\mu$ the dynamics of the associative recall with starting point x will converge to ξ_μ. The value $R=\min_\mu R_\mu$ is called the attraction radius of the network (AR).*

When inputs and memorized patterns belong to a normed vector space, if the additive noise does not exceed the attraction radius in this norm then all memories will be retrieved correctly during the associative recall.

The attraction radius can be estimated in the following special case:

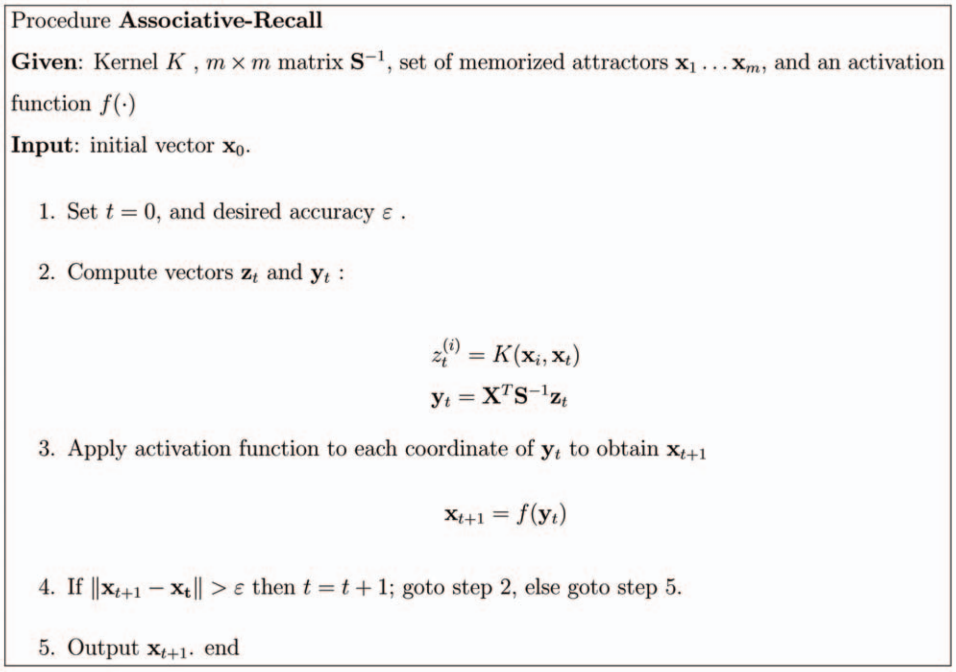
Theorem 3 *Suppose that a kernel associative memory has identity activation function in the output layer and a generalized sigmoid f in the hidden layer. Suppose also the kernel K satisfies the global Lipschitz condition on a certain domain $D\subset\mathbb{R}^n$, i.e., there exists L such that $\forall\mathbf{x}_1,\mathbf{x}_2,\mathbf{y}\in\|K(\mathbf{x}_1,\mathbf{y})-K(\mathbf{x}_2,\mathbf{y})|\le L\|\mathbf{x}_1-\mathbf{x}_2\|$. Then the stored patterns are attractors, and the memory attraction radius is*

$$R\geq\frac{c}{\|\mathbf{S}^{-1}\|\cdot L}$$

where c is a constant that depends only on f.

The proof of this theorem is given in Materials and Methods Sec. Proof of Theorem 3.

We further made a series of experiments of direct measurement of attraction radius for a dataset of 30×30 gray-scale face images. Results of this experiment are represented in Fig. 5.



```
Procedure Associative-Recall
Given: Kernel $K$ , $m \times m$ matrix $\mathbf{S}^{-1}$, set of memorized attractors $\mathbf{x}_1 \ldots \mathbf{x}_m$, and an activation function $f(\cdot)$
Input: initial vector $\mathbf{x}_0$.

  1. Set $t = 0$, and desired accuracy $\varepsilon$ .
  2. Compute vectors $\mathbf{z}_t$ and $\mathbf{y}_t$ :
         $z_t^{(i)} = K(\mathbf{x}_i, \mathbf{x}_t)$
         $\mathbf{y}_t = \mathbf{X}^T \mathbf{S}^{-1} \mathbf{z}_t$
  3. Apply activation function to each coordinate of $\mathbf{y}_t$ to obtain $\mathbf{x}_{t+1}$
         $\mathbf{x}_{t+1} = f(\mathbf{y}_t)$
  4. If $\|\mathbf{x}_{t+1} - \mathbf{x_t}\| > \varepsilon$ then $t = t + 1$; goto step 2, else goto step 5.
  5. Output $\mathbf{x}_{t+1}$. end
```

Figure 3. Algorithm of Associative Recall. An iterative procedure converges to an attractor.
doi:10.1371/journal.pone.0010955.g003

Remark 3 We have also proven that under the conditions of Theorem 3 all stored patterns are attractors. Yet, the memory was not proven to be free from spurious equilibria. However, spurious attractors were never observed in numerical experiments. The typical situation is that all the input domain is divided into attraction basins of the memorized vectors. The basins look like Voronoi polyhedra as depicted in Fig. 6.

2.3.2 Bounded Synaptic Weight Range. There is a connection between a bound on the values of the synapses (see [45]) and the kernel function defining the network.

Proposition 1 *Let the kernel memory have input data such that every two inputs* $\|\mathbf{x}_i - \mathbf{x}_j\| \geq \delta$, $\forall i,j = 1 \ldots m$, $i \neq j$ *and the piece-wise Mercer kernel* K *with this* δ *and certain* μ. *Then the synapses defined by matrix* $\mathbf{S}^{-1}$ *are bounded and the following bound holds:*

$$|\hat{s}_{i,j}| \leq \mu^{-1}$$

Proof. From the proof of Lemma 3.2 in Material and Methods we know that $\mathbf{S}$ could be written as $\mathbf{S} = \mathbf{S}' + \mu \mathbf{I}$, where $\mathbf{S}' \geq 0$ is a positive semidefinite matrix. By linear algebra we have $\|\mathbf{S}^{-1}\| \leq \mu^{-1}$. Finally $\|\mathbf{S}^{-1}\|_{\max} \equiv \max_{ij} |\hat{s}_{i,j}| \leq \|\mathbf{S}^{-1}\|$ by matrix norm equivalence in finite-dimensional space.

2.3.3 Maximizing Capacity. The kernel associative memory works as a symmetric network in an abstract feature space is used only in implicit way. Any implementation of the kernel associative memory with neural computational units requires a recurrent layered structure (Fig. 4). We can maximize the network capacity by using the approximation $\mathbf{S} \approx I$. This approximation is suitable if the stored patterns are sufficiently distant in the kernel view, see Remark 5. With this approximation one can save m^3 connections without significant loss of association quality by eliminating the middle layer in Fig 4a and the other two layers will have weight matrices $\mathbf{X}$ and $\mathbf{X}^T$, identical with respect to transposition; see Fig. 4b. So, to store m vectors of $\mathbb{R}^n$ we would need $m \times n$ real numbers only (lossless coding).

The definition of memory capacity and connections/neurons ratio now leads to

$$\frac{\max(m,n)}{\min(m,n)} \geq 1.$$

Remark 4 The approximation $\mathbf{S} \approx \mathbf{I}$ is suitable if the stored patterns are almost orthogonal in the feature space. For localized kernels (e.g. RBF) this means that the patterns are distant enough from each other (comparing to the characteristic scale of the kernel). Because the condition of orthogonality is applied in the feature space, not in the input space, this condition does not imply any relative size of m versus n.

With this optimization the kernel-memory network can be made arbitrarily sparse by choosing a sparse kernel, i.e., a kernel that explicitly depends only on a (small) portion of the coordinates of its argument vectors. The non-zero weights will correspond to the inputs that the sparse kernel depends on. The attractors will have a sparse structure analogous to the kernel as well. If our goal is to memorize arbitrary dense data we can still use the sparse network as long as encoding and decoding layers are externally added to it.

2.4 Flexibility of the Memory

2.4.1 Flexibility in the Attractor Space. The kernel associative memory can be made capable of adding and removing attractors explicitly.

To add a new attractor to the network we create a new neuron in the $\mathbf{S}$ matrix layer. The dimension of matrix $\mathbf{S}$ is increased from m to $m+1$. To do so we compute $s_{i,m+1} = K(\mathbf{x}_i, \mathbf{x}_{m+1})$ and we update the inverse $\mathbf{S}^{-1}$ efficiently using the linear-algebra identity [46]:

$$(\mathbf{A} + \mathbf{B})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}(\mathbf{I} + \mathbf{A}^{-1}\mathbf{B})\mathbf{A}^{-1} \qquad (16)$$

where $\mathbf{A} = \begin{pmatrix} \mathbf{S}_m & \mathbf{0} \\ \mathbf{0} & s_{m+1,m+1} \end{pmatrix}$ and

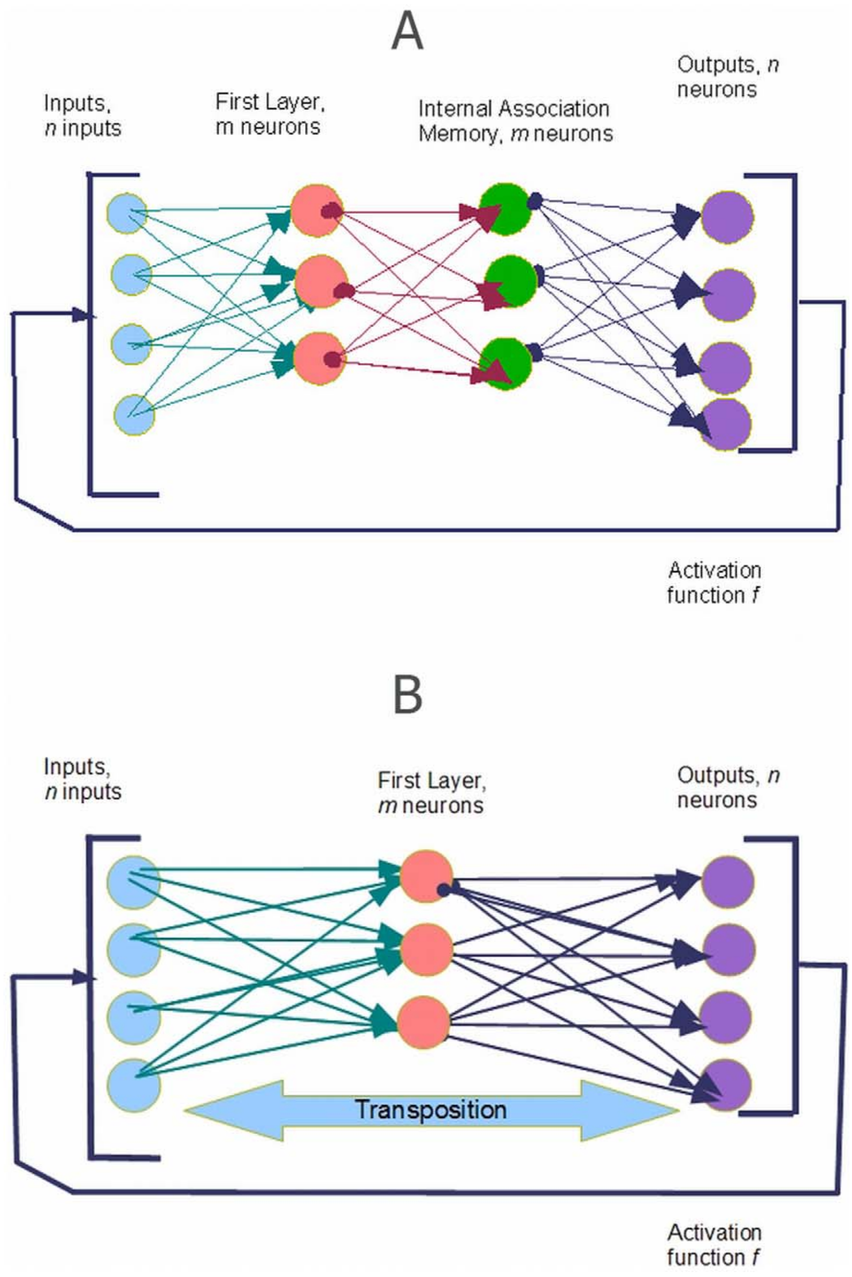


Figure 4. A possible neural-network representation of the Kernel memory. Architecture (a) corresponds to the algorithm of learning and recall as described in the text. In (b) we use an approximation to maximize the capacity and reduce the number of neural units. Middle layer with synaptic matrix $\mathbf{S}^{-1}$ is eliminated, synaptic matrices of the resulting two layers are identical up to transposition. Therefore we have $m \times n$ distinct connections (m is the number of stored memories and n is the input dimension). We can choose a ''primary'' layer to store the connections, the other one will mirror them. Effective memory capacity of this architecture is $\frac{\max(m,n)}{\min(m,n)}$.
doi:10.1371/journal.pone.0010955.g004

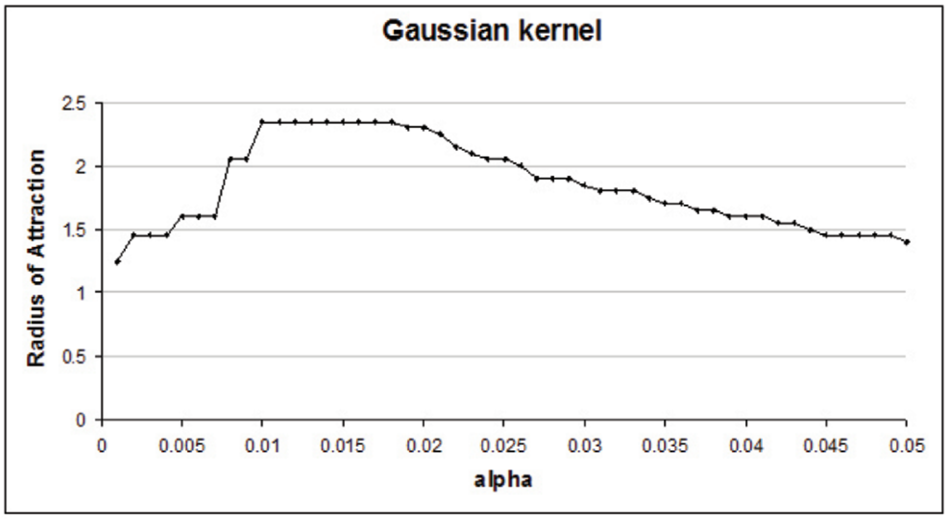
$$\mathbf{B} = \begin{pmatrix} & & & s_{1,m+1} \\ \mathbf{0}_{m\times m} & & & \vdots \\ & & & s_{m,m+1} \\ s_{m+1,1} & \dots & s_{m+1,m} & 0 \end{pmatrix}$$

Calculation using (16) takes $O(m^2)$ operations since $\mathbf{S}^{-1}$ is already known.

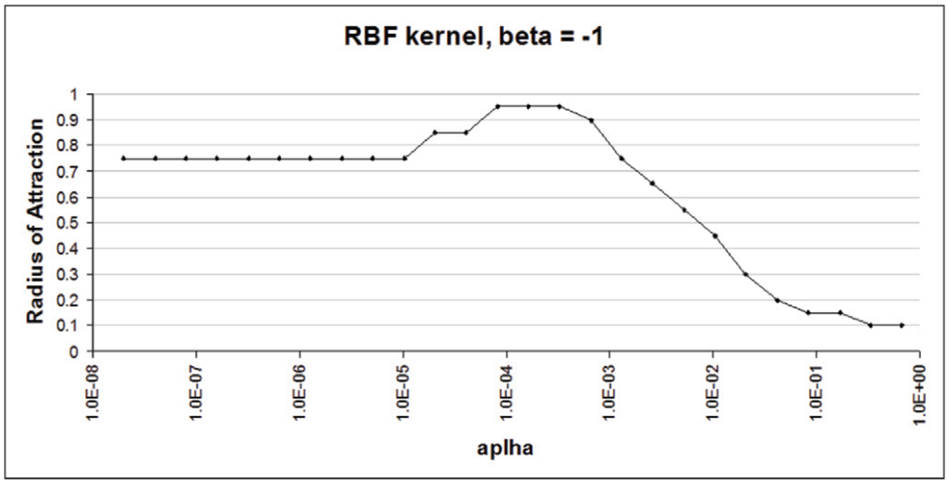
Similarly one can delete an attractor by reducing the dimension of $\mathbf{S}$. Here

$$\mathbf{A}+\mathbf{B} = \begin{pmatrix} \mathbf{S}_{m-1} & \mathbf{0} \\ \mathbf{0} & s_{m,m} \end{pmatrix} \mathbf{A} = \mathbf{S}_m$$

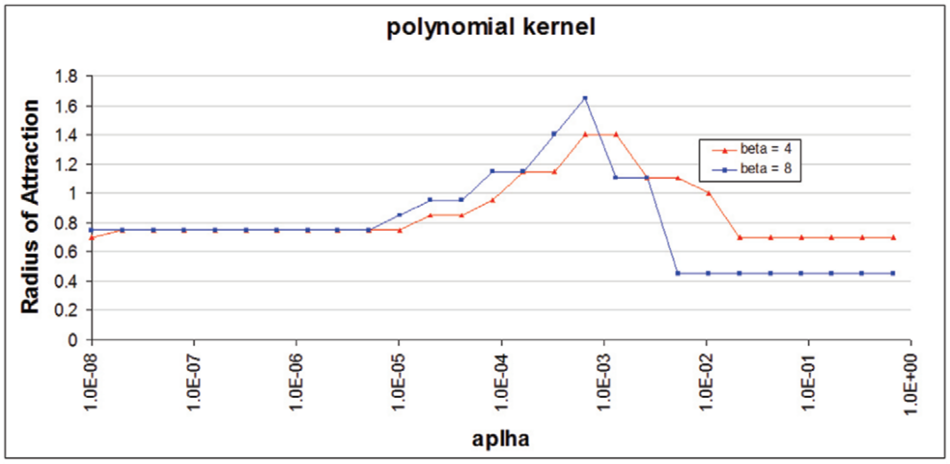
$$\mathbf{B} = \begin{pmatrix} & & & -s_{1,m} \\ \mathbf{0}_{(m-1)\times(m-1)} & & & \vdots \\ & & & -s_{m-1,m} \\ -s_{m,1} & \dots & -s_{m,m-1} & 0 \end{pmatrix}$$



(a) Gaussian Kernel



(b) Power RBF kernel



(c) Polynomial Kernel

Figure 5. Direct measurement of kernel memory's attraction radius. The dataset of 60 gray-scale 30×30 face images was used. They were normalized component wise to the segment $[-1;1]$. The graphs show dependence of the AR on kernel parameter α (for fixed β) for 3 standard types of kernel: Gaussian $K((x),(y))=\exp(-\alpha\|x-y\|^2)$ (a); power RBF: $K((x),(y))=(1+\alpha\|x-y\|^{\beta})$, $\alpha,\beta>0$ (b), and polynomial: $K(x,y)=(1+\alpha\langle\mathbf{x},\mathbf{y}\rangle)^{\beta}$, $\alpha>0$, β positive integer (c). The AR is relatively large,15–20% of memories' norm, and there exist an optimal value of α for each kernel.
doi:10.1371/journal.pone.0010955.g005

To receive $\mathbf{S}_{m-1}$, the last column and last row from matrix $\mathbf{A}+\mathbf{B}$ will be removed.

This results in the two algorithms of Fig. 7.

Remark 5 In the case where $\mathbf{S}$ is approximately a diagonal matrix, its inverse can be calculated by the approximation $\mathbf{S}=\lambda\mathbf{I}+\varepsilon\mathbf{S}_1$ for small ε, and $\mathbf{S}\approx\lambda\mathbf{I}$, which does not change during updates.

Remark 6 The procedure Add-Attractor (see Fig. 7) is local in time. To store a new pattern in the memory we only have to know the new attractor and the current connection matrix $\mathbf{S}^{-1}$.

In Fig. 6 we display an example of adding an attractor.

2.4.2 Flexibility in Input and Feature Spaces. External inputs may come with more or fewer features than previously, causing the input dimensionality to change with time. We propose a mechanism that enables the network to handle such heterogeneity of dimension with no need to relearn the previously learned inputs.

Assume that the current dimension in the input space consists of the "initial dimension" n and "new" q dimensions; denote this as $\mathbf{x}=(\mathbf{x}_a;\mathbf{x}_b)$. We will allow the change of dimensionality by changing the kernel itself: from the kernel K_n that considers the first n dimensions to kernel K_{n+q} that depends on all dimensions.

The change of kernel requires the recalculation of $\mathbf{S}^{-1}$. However, this need not require $O(m^3)$ operations if we constrain to kernels that can be written in an additive form:

$$\begin{aligned}K_{n+q}(\mathbf{x},\mathbf{y})=K_n(\mathbf{x}_a,\mathbf{y}_a)+K_q(\mathbf{x}_b,\mathbf{y}_b)\\+K_{int}(\mathbf{x}_a,\mathbf{y}_b)+K^*_{int}(\mathbf{x}_b,\mathbf{y}_a)\end{aligned}\tag{17}$$

where K_{int} describes the interaction of n and q. An explicit kernel with this property is the polynomial kernel (see Section "Variable Kernel and Dimensionality"). Algorithms for dimensionality control appear in Fig. 8. An example is given in Example 3 in the next section.

We also prove Lemma 4 in Materials and Methods Section "Variable Kernel and Dimensionality", stating that a small alteration to the kernels enables changing input dimensionality without loosing previously learnt attractors.

2.5 Memory Consolidation and Clustering

The memory system with its loading algorithm enables consolidation of inputs into clusters using the competitive learning method. Suppose we have a learning sample of N vectors $\mathbf{x}_1,\ldots\mathbf{x}_N$ and m clusters have to be created. Random vectors initiate the m attractors. When a new input is provided, the recall procedure is performed and the attractor of convergence $\mathbf{x}^*_k$ is updated by $\mathbf{x}^*_k\leftarrow(1-\alpha_v)\mathbf{x}^*_k+\alpha_v\mathbf{x}_v$. Parameter sequence α_v is selected in order to provide better convergence of attractors: for instance, we can take α_v such that $\lim_{v\to\infty}\alpha_v=0$ but $\sum_v\alpha_v\to\infty$. This step is repeated until all attractors stabilize.

We tested the consolidation algorithm using the MNIST database of handwritten digits [47]. The data consists of ten different classes of grayscale images (from '0' to '9', each of 28×28 pixels in size) together with their class labels.

Experiment 1: *Clustering with the Kernel Memory.* The goal of this experiment is to demonstrate performance of memory clustering. For this purpose memory was trained on the learning sample in order to form attractors. Then attractors were assigned to classes, and classification rate was measured on an independent test sample.

For the MNIST data we used principal-component (PC) preprocessing. We took the first $d=200$ PCs which contain 96.77% of the variance. The learning sample contained 10000 digits, 1000 from each class. The kernel was chosen in the form:

$$K(\mathbf{x},\mathbf{y})=\exp\left(\frac{-\alpha}{2R^2}\sum_{k=1}^{d}w_k(x_k-y_k)^2\right)\tag{18}$$

where $R^2=\sum_{k=1}^{d}w_k$, and α is a parameter. This is a Gaussian kernel depending on weighted metric. Weights were chosen as:

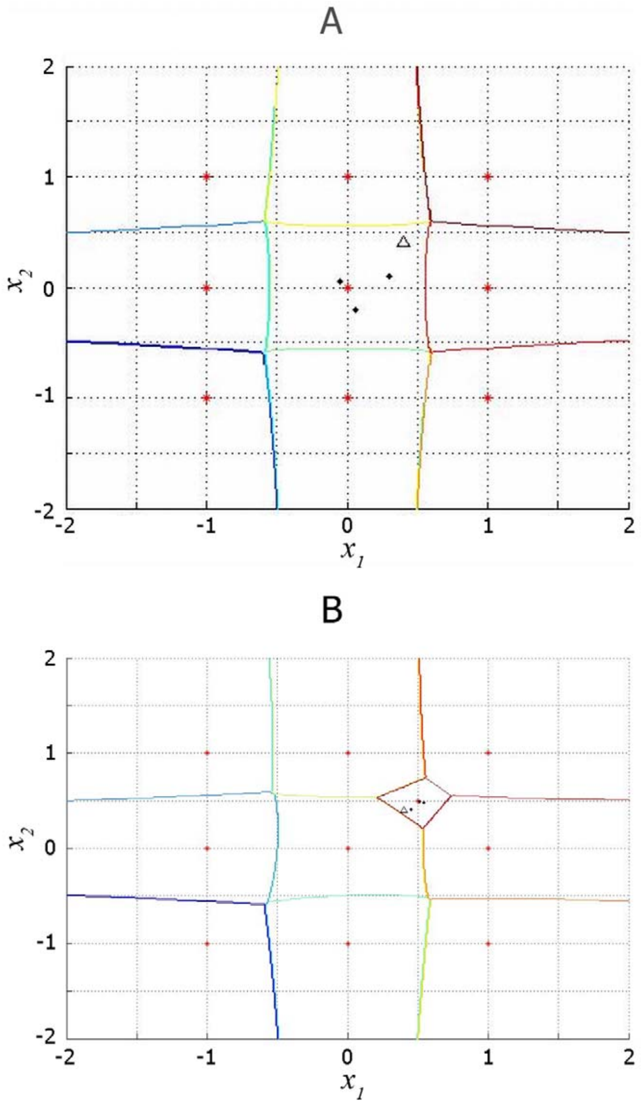


Figure 6. Example of an explicit insertion of an attractor. (a): 9 attractors in 2D (red stars), attraction basins are bounded by Voronoi polygons. (b): A tenth attractor (0.50.5) is added. Trajectories of the associative recall dynamics starting at (0.4, 0.4) (triangle) are shown in black dots. The destination attractor for this initial condition has been changed since the new attractor was stored.
doi:10.1371/journal.pone.0010955.g006

```
Procedure **Add-Attractor:**
**Given:** Kernel $K$ , $m\times m$ matrix $\mathbf{S}_m^{-1}$, set of memorized attractors $\mathbf{x}_1\ldots\mathbf{x}_m$.
    1. Set $m \leftarrow m+1$
    2. Update $\mathbf{S}^{-1}$ using identity (18)
    3. Store vector $\mathbf{x}_{m+1}$ in $\mathbf{X}$ for calculation of $z_{m+1}$ during recall.
```

```
Procedure **Remove-Attractor:**
**Given:** Kernel $K$ , $m\times m$ matrix $\mathbf{S}_m^{-1}$, set of memorized attractors $\mathbf{x}_1\ldots\mathbf{x}_m$.
    1. Set $m \leftarrow m-1$
    2. Update $\mathbf{S}^{-1}$ using identity (18)
    3. Remove vector $\mathbf{x}_m$ from $\mathbf{X}$.
```

Figure 7. Procedures of adding and removing attractors in kernel autoassociative memory.
doi:10.1371/journal.pone.0010955.g007

$$w_k=\left(\frac{STD(x_k)}{\frac{1}{Q}\sum_{l=1}^{Q}STD_l(x_k)}-1\right)^2 \tag{19}$$

We also tried the formula:

$$w_k=\frac{1}{Q}\sum_{l=1}^{Q}(E_l x_k-Ex_k)^2 \tag{20}$$

where E_l and STD_l are expectation and standard deviation over the l-th class, and Q is the quantity of classes. However formula (19) gave better results.

Because of the complexity of the MNIST data, we chose to have multiple clusters per class. Table 1 summarizes the classification rates for different amounts of attractors in the memory. The classification is slightly superior to other unsupervised clustering techniques (even that the goal of the memory system is not directly in clustering). The number of memory attractors required for good clustering is also smaller than other techniques, e.g. [48]. Figure 9.a) provides an example of typical memory attractors of each class.

We also made a series of experiments with $w_k=\frac{\sigma_k}{R}$; $R=\sqrt{\sum\sigma_k^2}$, where σ_k is an STD of k-th principal component. This weighting does not depend on class labels in any way. We can see (last row of table 1) that results are poor for small number of attractors per class, but for higher number of attractors classification rate is even better.

Experiment 2. *Clustering under changing input dimensionality*. This experiment demonstrates clustering while input dimensionality increases and the kernel is being changed. For this purpose, the resolution of the original images was reduced twice, to 14×14 (Fig. 9). Then the images were passed through a linear transformation in order to use the kernel (19). The memory was trained on 10,000 such digit images, forming 100 attractors. The recognition quality obtained was 76.4%. Then the kernel was extended in order to work with the original size (28×28), and another 10,000 digits were added, now in full-size. This second session of learning started from the previous set of attractors, without retraining. The final classification rate was enhanced to 85.4%.

Experiment 3. *Explicit example of adding input dimension*. Consider the $\mathbb{R}^2$ data where points lie on two Archimedes' spirals:

$$\begin{aligned}x_1&=k\varphi\cos(\varphi)\\x_2&=k\varphi\sin(\varphi)\end{aligned} \tag{21}$$

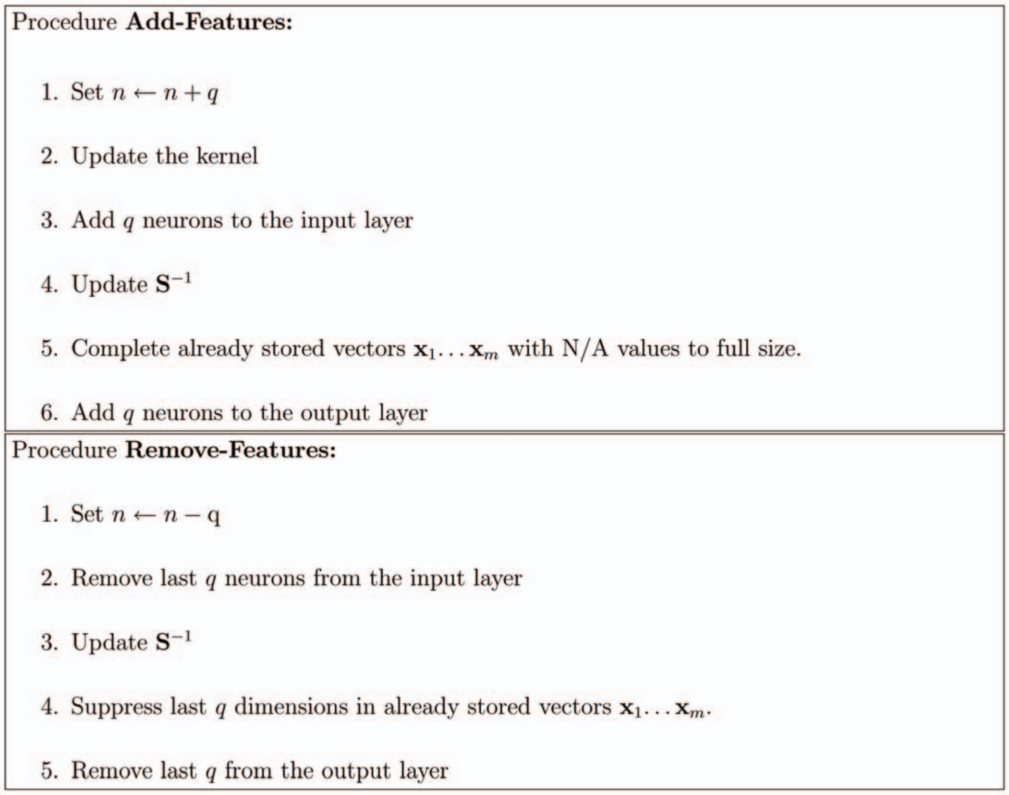
and

$$\begin{aligned}x_1&=k\varphi\cos(\varphi+\pi)\\x_2&=k\varphi\sin(\varphi+\pi)\end{aligned} \tag{22}$$

We chose angle range $\varphi\in[\pi/5:8\pi]$ for both classes. The initial kernel was $K_2=\exp(-\frac{(u_1-v_1)^2+(u_2-v_2)^2}{2\sigma_2^2})$. Then we add 3-d coordinate $x_3=\lambda_{1,2}\sqrt{x_1^2+x_2^2}$, where $\lambda_1=1$ and $\lambda_2=1.5$ for first and second class. For $\mathbb{R}^3$ the additive kernel will be $K_3=K_2+K_1$, where $K_1=\exp(-\frac{(u_3-v_3)^2}{2\sigma_1^2})$, an interaction term is not necessary in this example. We took $\sigma_1=0.25$ and $\sigma_2=0.2$. At first, the network was loaded with the 40 data points in $\mathbb{R}^2$. Each point was labeled and a classification was executed. The recognition quality on an independent test sample was 86.4%. Then the training was continued with the additional 40 inputs in $\mathbb{R}^3$ and the final classification rate increased to 97.5%.

2.6 Synaptic Plasticity and Memory Reconsolidation

Reconsolidation is a storage process distinct from the one time loading by consolidation. It serves to maintain, strengthen and modify existing memories shortly after their retrieval [49]. Being a key process in learning and adaptive knowledge, problems in reconsolidation have been implicated in disorders such as Post



```
Procedure Add-Features:
    1. Set $n \leftarrow n + q$
    2. Update the kernel
    3. Add $q$ neurons to the input layer
    4. Update $\mathbf{S}^{-1}$
    5. Complete already stored vectors $\mathbf{x}_1 \ldots \mathbf{x}_m$ with N/A values to full size.
    6. Add $q$ neurons to the output layer

Procedure Remove-Features:
    1. Set $n \leftarrow n - \mathrm{q}$
    2. Remove last $q$ neurons from the input layer
    3. Update $\mathbf{S}^{-1}$
    4. Suppress last $q$ dimensions in already stored vectors $\mathbf{x}_1 \ldots \mathbf{x}_m$.
    5. Remove last $q$ from the output layer
```

Figure 8. Algorithms assuring flexibility in the input dimension.
doi:10.1371/journal.pone.0010955.g008

Traumatic Stress disorder (PTSD), Obsessive Compulsive disorder (OCD), and even addiction. Part of the recent growing interest in the reconsolidation process is the hope that controlling it may assist in psychiatric disorders such as PTSD [50] or in permanent extinction of fears [51].

2.6.1 Current Model of Reconsolidation in Hopfield Networks. A model of reconsolidation was introduced in [19]. It contains a learning mechanism that involves novelty-facilitated modifications, accentuating synaptic changes proportionally to the difference between network input and stored memories. The formula updating the weight matrix **C** is based on the Hebbian rule:

$$\mathbf{C}_{t+1} = (1 - w_\mu)\mathbf{C}_t + (w_\mu + \eta H)\mathbf{x}_t \mathbf{x}_t^T \qquad (23)$$

Here t is the time of the reconsolidation process, w_μ is a weight parameter defining learning rate, $\mathbf{x}_t$ is the current input stimulus, H is a Hamming distance from $\mathbf{x}_t$ to the set of network's attractors, and η is the sensitivity to the novelty of stimuli. This formula differs from the original Hebbian rule by having both weight decay and Hamming-distance terms affecting the learning.

Table 1. Experiment 3.

Attractors per class	1	10	20	50	100
Classification rate, %	52.4	80.4	82.2	87.8	91.1
Classification rate, $w_k = \sigma_k / R$, %	34.5	74.48	82.3	89.09	91.38

MNIST digits clustering. Classification rate vs. number of concepts.
doi:10.1371/journal.pone.0010955.t001

The model predicts that memory representations should be sensitive to learning order.

2.6.2 Our Reconsolidation Algorithm. In the case of Hebbian learning, the network's synaptic matrix is composed of a linear space. In our kernel associative memory, on the other hand, the corresponding space is no longer linear but rather is a Riemannian manifold, see Materials and Methods Section 3.7. Additions and multiplications by a scalar are not defined in this space and thus formula (23) cannot no longer be applied.

To remedy the situation we define a Riemannian distance (see Material and Methods Section 3.7) and a geodesics which enables the memory to change gradually as new but close stimuli arrive [52]: a point on a geodesic between x_1 and x_2 that divides the path in ratio $\alpha/(1-\alpha)$ is a generalization of the convex combination $\alpha x_1 + (1-\alpha)x_2$. Suppose, initially we have a memory $\mathbf{X}_0$ that contains m attractors $x_{0,1}, x_{0,2} \ldots x_{0,m}$. Then we obtain $\mathbf{X}_1$ by replacing one attractor by a new stimulus: $x_{0,1} \rightarrow x_{1,1}$. The distance between $\mathbf{X}_0$ and $\mathbf{X}_1$ can be thought of as a measure of "surprise" that the memory experience when meets new stimuli. To track the changes, the memory moves slightly on the manifold from $\mathbf{X}_0$ to $\mathbf{X}_1$. See algorithm in Fig. 10.

2.6.3 Numerical Experiments. We exemplify the power enabled to us by the reconsolidation with the following experiments.

Experiment 4. *Morphed faces*. The goal of this experiment is both to show the performance of the reconsolidation process we describe on large-scale data and to compare its properties with the recent psychological study [35].

Morphed faces were created by Joshua Goh at the University of Illinois. The faces were obtained from the Productive Aging Lab Face Database [53] and other volunteers (All the face images used in our research were taken from the Productive Aging Lab's Face

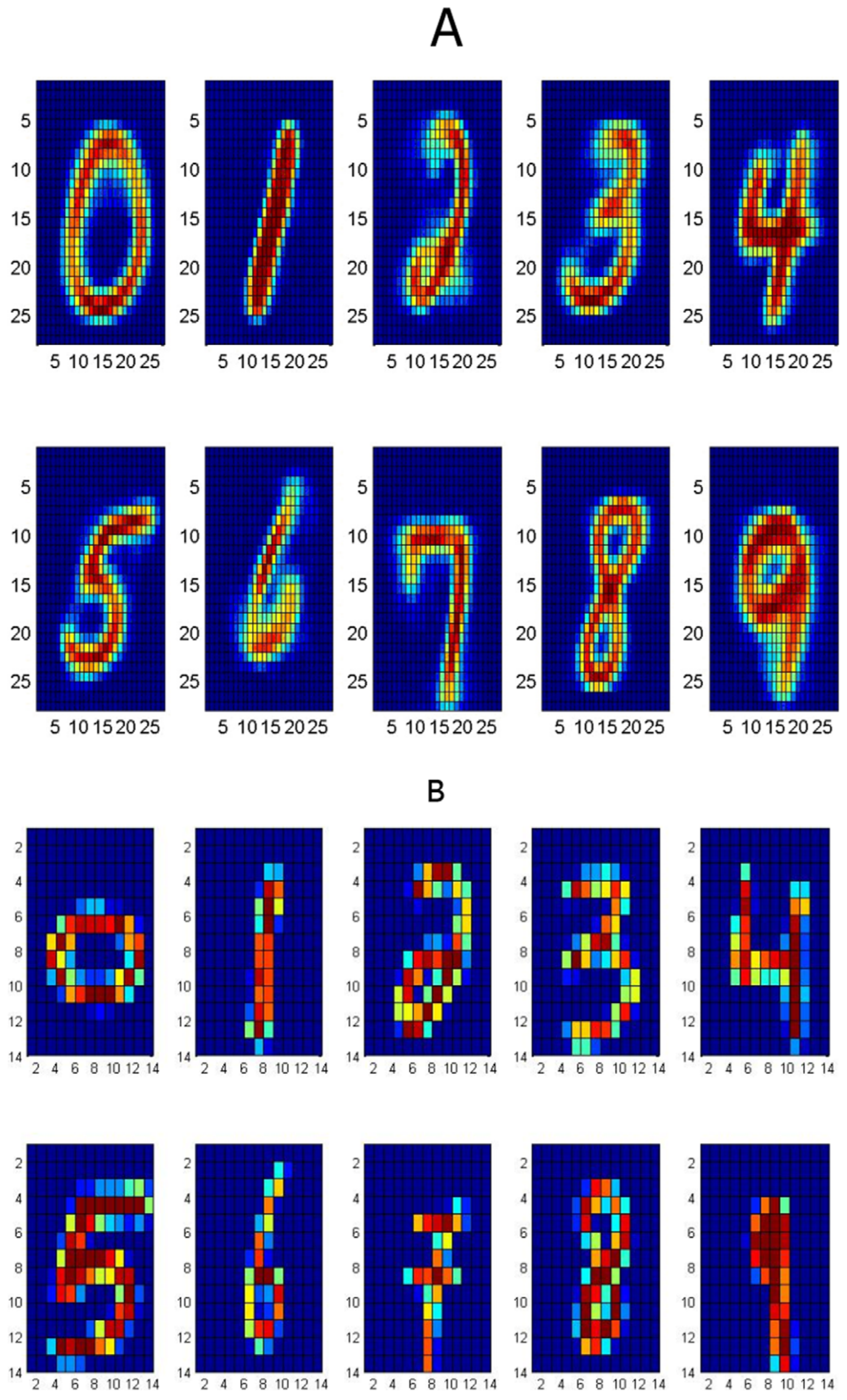


Figure 9. The MNIST experiment. (a): Ten typical attractors, one for each class out of 100 attractors. (b): An example of 10 downscaled digits. Each digit in (b) is a $14 \times 14 = 196$-dimensional vector. Experiment 2 demonstrates work of the algorithm of adding features. we started learning with 10000 downscaled images. To continue training we used another 10000 images, and we had to add $784–196 = 588$ features.
doi:10.1371/journal.pone.0010955.g009

Database, Morphed face dataset. This dataset is freely accessible from https://pal.utdallas.edu/facedb/request/index/Morph). They contain a mix of young, old, Asian, Western, male, and female faces. They are gray-scale with luminance histogram equated. Faces were morphed using the software Sqirlz morph. Original size of all images was 640×480. Useful area falls in the

```
Procedure Geodesic-Update
Given: Current state of the memory $\mathbf{X}_t^0$, parameter $\alpha \in [0;\ 1]$
Input: Current stimulus $\mathbf{x}_t$, time $t$.

    1. Run Associative-Recall of a memory $\mathbf{X}_t^0$ on input $\mathbf{x}_t$, denote the recall by $\mathbf{x}_t'$
    2. Remove-Attractor $\mathbf{x}_t'$ from $\mathbf{X}_t^0$
    3. Add-Attractor $\mathbf{x}_t$ to the memory. Denote the result as $\mathbf{X}_t^1$
    4. Build a geodesic $\gamma$ between $\mathbf{X}_t^0$ and $\mathbf{X}_t^1$. Denote its length by $L$.
    5. Take a point $\mathbf{X}_t^*$ on $\gamma$ in distance $\alpha L$ from $\mathbf{X}_t^0$.
    6. Set time $t+1$ Output $\mathbf{X}^{t+1} = \mathbf{X}_t^*$ .
```

Figure 10. Algorithm of Geodesic Update.
doi:10.1371/journal.pone.0010955.g010

rectangle 320×240, images were cropped to this size before entering to the network. The database contains 150 morph sequences, each of them consists of 100 images.

In our simulations we created a network with 16 attractors representing 16 different faces; it had 76800 input and output neurons, and two middle layers of 16 neurons each. Four arbitrarily selected network's attractors are depicted in Fig. 11. A Gaussian kernel was chosen in order to simplify calculations with large scale data.

Attractors were initialized with first images from 16 arbitrarily selected morph sequences. When the learning order followed image order in the morphing sequence, attractors changed gradually and consistently. The ability to recognize the initial set of images gradually decreased when attractors tended to the final set. In case of random learning order attractors quickly became senseless, and the network was not able to distinguish faces.

This experiment generalizes the result shown in [19] but is done on real images demonstrating the efficiency of the reconsolidation process in kernel memories for high dimension and multi-scale data. In accordance with [35], the formation of "good" dynamic attractors occurred only when morphed faces were presented in order of increasing distance from the source image. Also, as shared also with [34,19]: the magnitude of the synaptic update due to exposure to a stimulus depends not only on the current stimulus (as in Hebbian learning) but also on the previous experience, captured by the existing memory representation.

Experiment 5. *Tracking Head Movement.* This example focuses on rotating head images for reconsolidation based on the VidTIMIT dataset [54], and it demonstrates our algorithm on a more applied example of faces and computer vision. The VidTIMIT dataset is comprised of video and corresponding audio recordings of 43 people. The recordings include head rotation sequences. The recording was done in an office environment using a broadcast quality digital video camera. The video of each person is stored as a numbered sequence of JPEG images with a resolution of 512×384 pixels.

The ability to track and recognize faces was tested on the sets of 15 last frames from each sequence. Example of attractors during the reconsolidation in this experiments is depicted in the Fig. 12. With reconsolidation and ordered stimuli the obtained recognition rate was 95.2%. If inputs were shuffled randomly, attractors got messy after 30–50 updates, and the network did not demonstrate significant recognition ability.

Experiment 6. *Tracking The Patriot Missiles.* The following experiment takes the reconsolidation model into a practical technology that follows trajectories in real time in complex dynamic environments.

We analyzed videos of Patriot missile launches with resolution 320×240, originally in RGB color, and transformed them to grayscale. The memory was loaded with vector composed of two 40×40-pixel regions (windows) around the missile taken from two consequent frames and a two-dimensional shift vector indicating how the missile center has moved between these frames. Optimal number of attractors was found to be 16–20.

Using memory reconsolidation algorithm we were able to calculate velocity vector every time, and therefore track the missile with great precision, with only average error of 5.2 pixels (see Fig. 13).

2.7 Conclusions

We have proposed a novel framework of kernel associative memory as an attractor neural network with a high degree of flexibility. It has no explicit limitation, either on the number of stored concepts or on the underlying metric for association, since the metric is based on kernel functions. Kernels can be slightly changed as needed during memory loading without damaging existing memories. Also due to the kernel properties, the input dimension does not have to be fixed. Unlike most other associative memories our model can both store real -valued patterns and allow for analogous attractor-based dynamic associative recall.

We endowed our memory with a set of algorithms that insure flexibility, enabling it to add/delete attractors as well as features (dimensions) without need to retrain the network. Current implementation of our memory is based on a simple competitive clustering algorithm and consolidates memories in a spirit similar to the localist attractor networks [43]. We have experimentally tested the memory algorithms on the MNIST database of handwritten digits, a common benchmark for learning machines. The obtained clustering rate for this database (91.2%) is slightly better than the best known result for unsupervised learning

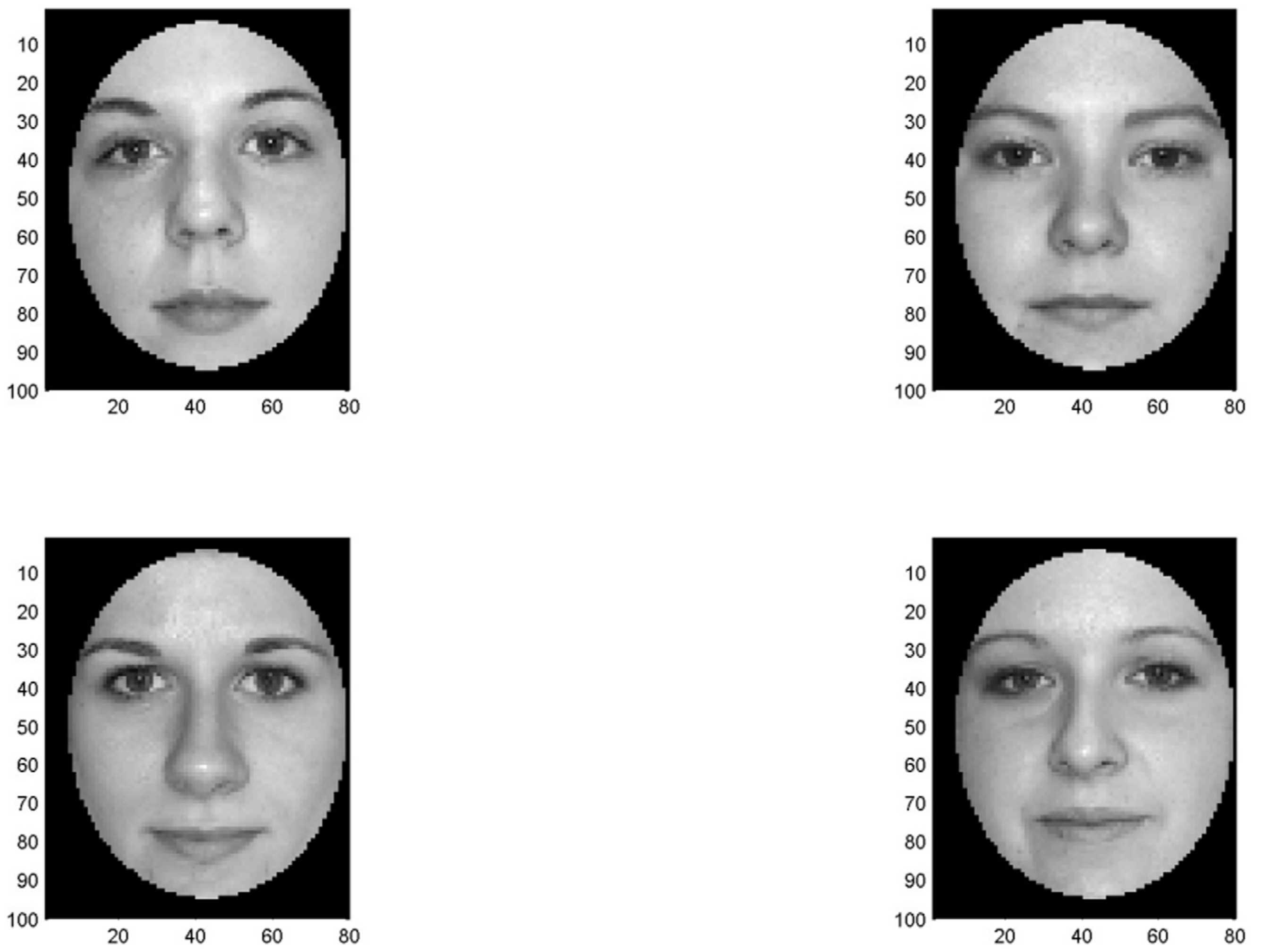


Figure 11. Example of attractors during face reconsolidation.
doi:10.1371/journal.pone.0010955.g011

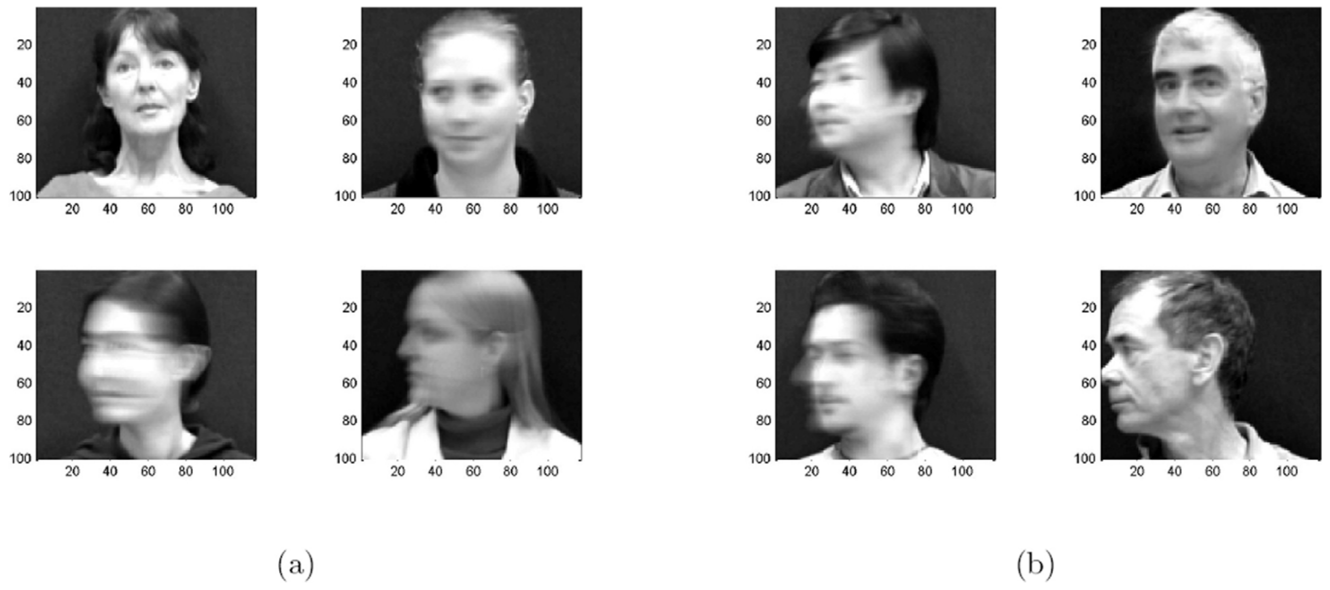


Figure 12. Tracking rotating heads via reconsolidation. Memory attractors are blurred when motion is quick.
doi:10.1371/journal.pone.0010955.g012

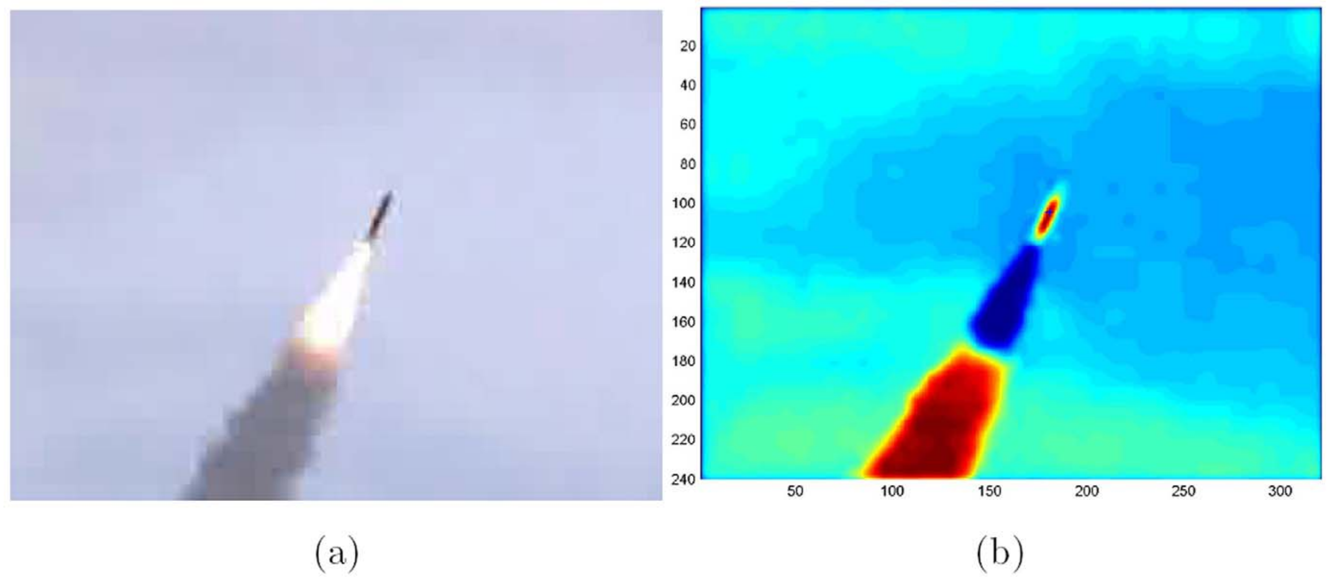


Figure 13. Patriot missile example. Original frame (a) and a processed frame with tracking marker (b).
doi:10.1371/journal.pone.0010955.g013

algorithms on this benchmark. The model further allows the process of reconsolidation after memory is stored when retrieval by similar patterns is activated. We demonstrated the properties of reconsolidation on gray scale large image faces in morphing experiments. Based on the theoretical and experimental research made in the present paper we conclude that the proposed kernel associative memory is promising both as a biological model and a computational method.

Materials and Methods

3.1 Piece-wise Mercer Kernels

The classical Kernels $K(x,y)$ introduced by Vapnik to the field of Machine Learning had the Mercer condition. That is, for all square integrable functions $g(x)$ the kernel satisfied:

$$\iint K(x,y)g(x)g(y) \geq 0. \tag{24}$$

The Mercer theorem states that if K satisfies the Mercer condition there exists a Hilbert space H with basis $\mathbf{e}_1,\mathbf{e}_2,\ldots\mathbf{e}_n\ldots$ and a function φ: $E_x \to H$, $\varphi(\mathbf{x}) = \sum_{k=1}^{\infty} \varphi_k \mathbf{e}_k$, where $\varphi_1,\varphi_2\ldots : E_x \to \mathbb{R}$, such that

$$K(\mathbf{u},\mathbf{v}) = \sum_{n=1}^{\infty} a_n \varphi_n(\mathbf{u}) \varphi_{\mathrm{n}}(\mathbf{v}) \tag{25}$$

and all $a_n > 0$. That is, K is a scalar product of $\varphi(\mathbf{u})$ and $\varphi(\mathbf{v})$

General Mercer kernels are not sufficient for creating the associative memory since our kernel memories require that all attractors are linearly independent in the feature space, and some Mercer kernels do not assure it, such as the basic scalar-product kernel $K(\mathbf{u},\mathbf{v}) = <\mathbf{u},\mathbf{v}>$. As shown in Lemma 2, linear independence of attractors in the feature space is needed to provide correct association in our system. The piece-wise Mercer kernels as we define next have this desired property.

Definition 3 *A kernel $K(\boldsymbol{u},\boldsymbol{v})$ is said to satisfy the piece-wise Mercer condition if there exist $\delta > 0$, $\mu > 0$, and a Mercer kernel $K'(\mathbf{u},\mathbf{v})$ such that:*

$$K'(\mathbf{u},\mathbf{v}) = \begin{cases} K(\mathbf{u},\mathbf{v}), & \|\mathbf{u}-\mathbf{v}\| \geq \delta \\ K(\mathbf{u},\mathbf{v}) - \mu, & \mathbf{u} = \mathbf{v} \end{cases} \tag{26}$$

and

$$K'(\mathbf{u},\mathbf{v}) \leq K(\mathbf{u},\mathbf{v}),\ \|\mathbf{u}-\mathbf{v}\| \leq \delta. \tag{27}$$

The piece-wise Mercer kernel is an extension of strictly positive definite (SPD) kernels. These kernels have some internal property of regularization. For usual Mercer kernels, e.g. common scalar product there are still situations when Gram matrix of certain vector set in the feature space will be degenerate. In contrast to this, the $\mu-\delta$ piece-wise Mercer kernel can guarantee that for any finite set of vectors their Gramian will be full-rank and even greater than $\mu\mathbf{I}$.

Lemma 1 *If K is a piece-wise Mercer Kernel, it also satisfies the Mercer condition.*

Proof. Indeed, for all square integrable functions $g(\mathbf{x}) : E_x \to \mathbb{R}$, the Mercer condition and inequality (26–27) give:

$$\iint \mathbf{K}(\mathbf{u},\mathbf{v}) g(\mathbf{u}) g(\mathbf{v}) d\mathbf{u} d\mathbf{v} \geq \iint \mathbf{K}'(\mathbf{u},\mathbf{v}) g(\mathbf{u}) g(\mathbf{v}) d\mathbf{u} d\mathbf{v} \geq 0$$

and the Mercer condition for K is fulfilled.

Remark 7 There is following relation between our definition of piece-wise Mercer kernel and standard notion of strictly positive definite (SPD) kernel (which is formulated by replacing "$\geq$" with "$>$" in (24): any piece-wise Mercer kernel is also SPD and for any continuous SPD kernel K there exist certain μ and δ such that K is $\mu-\delta$ piece-wise Mercer.

Example 1 *We demonstrate how to build a piece-wise Mercer kernel that would fit a given sample to be loaded in memory. Consider Gaussian Kernel* $K(\mathbf{u},\mathbf{v}) = \exp(-\frac{(\mathbf{u}-\mathbf{v})^2}{2\sigma^2}) \equiv R(\|\mathbf{u}-\mathbf{v}\|)$. *We can construct the kernel:*

$$K'(\mathbf{u},\mathbf{v}) = \begin{cases} 1-\mu-\dfrac{(1-\mu-R(\delta))\|\mathbf{u}-\mathbf{v}\|}{\delta}, & \|\mathbf{u}-\mathbf{v}\| \le \delta \\ K(\mathbf{u},\mathbf{v}), & \|\mathbf{u}-\mathbf{v}\| > \delta \end{cases} \tag{28}$$

The kernel K' *is constructed in that way that it is continuous and convex as a function of* $\|\mathbf{u}-\mathbf{v}\|$ *for any* μ *and* δ *such that* $1-\mu-R(\delta)>0$. *According to the Polya criterion (see, [37], sec. 10.8.4) we get that* K' *fulfills the Mercer condition. This shows that Gaussian kernel* K *is a piece-wise Mercer kernel.*

3.2 Correctness of Association

Lemma 2 *Suppose kernel* $K(\boldsymbol{u},\boldsymbol{v})$ *satisfies the piece-wise Mercer condition for certain* $\delta > 0$ *and there are input vectors* $\mathbf{x}_1\ \mathbf{x}_2 \ldots \mathbf{x}_m$ *such that* $\|\mathbf{x}_i - \mathbf{x}_j\| > \delta, \quad i \neq j$. *Then there exists a Hilbert space* H *and a nonlinear transformation* φ: $E_x \to H$ *such that*

1. $K(\mathbf{u},\mathbf{v}) = \langle \varphi(\mathbf{u}), \varphi(\mathbf{v}) \rangle_H$
2. $\varphi(\mathbf{x}_i), i=1\ldots m$ *are linearly independent.*

Proof. Since K is a Mercer kernel according to lemma 1, statement 1 is true by the Mercer theorem.

To prove 2) note that by definition 1 there exists a kernel K' that satisfies the inequality (26). K' is a Mercer kernel, and hence matrix $\mathbf{S}'$, $s'_{ij} = K'(\mathbf{x}_i,\mathbf{x}_j)$ would be non-negative definite. By our choice of K', $\mathbf{S} = \mathbf{S}' + \mu\mathbf{I}$ is strictly positive then invertible, as a sum of a positive-semidefinite and positive scalar matrices. That means that $\varphi(\mathbf{x}_i)$ are linearly independent in H, and we are done.

We use the following facts from linear algebra:

a. *Sum of a positive semidefinite matrix and positive scalar matrix is a strictly positive (symmetric) matrix*
b. *If matrix of pairwise scalar products of a finite set of vectors in Hilbert space is strictly positive the vectors are linearly independent.*

Lemma 3 *For any given finite learning sample* $\mathbf{x}_1 \ldots \mathbf{x}_m$ *of distinct vectors there exists a Kernel* K, *Hilbert space* H, *and a nonlinear transformation* φ: $E_x \to H$ *such that*

1. $K(\mathbf{u},\mathbf{v}) = \langle \varphi(\mathbf{u}), \varphi(\mathbf{v}) \rangle_H$
2. $\varphi(\mathbf{x}_i), i=1\ldots m$ *are linearly independent.*

Proof. Take $\delta = \frac{1}{2} \min_{i,j=1\ldots m,\ i\neq j} \|\mathbf{x}_i - \mathbf{x}_j\|$. Let us take Gaussian Kernel from Exapmple 1 with $\sigma = \delta/2$. As we have shown above in the Example 1, it will satisfy piece-wise Mercer condition with δ and any $\mu < 1-e^{-2}$. Then we apply Lemma 2 to K and we are done.

Next we provide the proof of Theorem 1.

Proof. Let $\delta = \frac{1}{2} \min_{i,j=1\ldots m,\ i\neq j} \|\mathbf{x}_i - \mathbf{x}_j\|$. Pick a kernel that satisfies piece-wise Mercer condition with this δ and certain $\mu>0$. Existence of at least one such a kernel for every δ is guaranteed by Example 1 (see MM) that shows how to construct piece-wise Gaussian kernels. Then, the conditions of Lemma 2 are fulfilled by kernel K and input set $\mathbf{x}_i$. Therefore, $\mathbf{S}$, is invertible, $\varphi(\mathbf{x}_i)$ are independent in the feature space, and association (9) — (11) is well defined.

3.3 Example of the associative recall

Example 2 ***Autoassociative memory***. *Let* $E_x = \mathbb{R}^3$, *and* $E_\vartheta = H = \mathbb{R}^6$. *Take,*

$$\varphi(\mathbf{x}) \equiv \varphi\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} x_2 \\ x_3 \\ x_1x_2 \\ x_1x_3 \\ x_2x_3 \end{pmatrix}$$

The kernel will be

$$K(\mathbf{u},\mathbf{v}) = <\varphi(\mathbf{u}),\varphi(\mathbf{v})> =$$

$$= u_1v_1 + u_2v_2 + u_3v_3 + u_1v_1u_2v_2 + u_1v_1u_3v_3 + u_2v_2u_3v_3$$

Take sigmoid activation function: $f(x) = \dfrac{\exp(10(x-0.5))}{1+\exp(10(x-0.5))}$

Suppose we have to memorize the following $m=5$ *vectors:*

$$\mathbf{x}_1 \ldots \mathbf{x}_5 = \begin{pmatrix} 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \end{pmatrix}$$

If we apply φ *to this set we have*

$$\varphi(\mathbf{x}_1..\mathbf{x}_5) = \begin{pmatrix} 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \end{pmatrix}$$

Vectors became linearly independent but their dimension has inflated. The matrix $\boldsymbol{S}$ *is:*

$$\mathbf{S} = \begin{pmatrix} 6 & 3 & 1 & 3 & 1 \\ 3 & 3 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 3 & 1 & 1 & 3 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

det($\boldsymbol{S}$*) = 4, we can compute its inverse:*

$$\mathbf{S}^{-1} = \begin{pmatrix} 0.75 & -0.5 & -0.25 & -0.5 & -0.25 \\ -0.5 & 1 & 0.5 & 0 & -0.5 \\ -0.25 & 0.5 & 1.75 & -0.5 & -0.25 \\ -0.5 & 0 & -0.5 & 1 & 0.5 \\ -0.25 & -0.5 & -0.25 & 0.5 & 1.75 \end{pmatrix}$$

Suppose the input vector for recall is $\mathbf{x}_0 = (0.22,\ 0.75,\ 0.8)^T$.

First iteration. *Starting from* $\boldsymbol{x}_0$ *compute* $z_0 = \begin{pmatrix} 2.535 \\ 1.02 \\ 0.75 \\ 2.15 \\ 0.22 \end{pmatrix}$

then $\mathbf{y}_0 = \begin{pmatrix} -0.28 \\ 0.25 \\ 0.20875 \end{pmatrix}$, *and* $\mathbf{x}_1 = \begin{pmatrix} 0.057324 \\ 0.92414 \\ 0.88968 \end{pmatrix}$.

Second iteration leads to $\mathbf{y}_1 = \begin{pmatrix} 0.057324 \\ 0.92414 \\ 0.85957 \end{pmatrix}$ *and* $\mathbf{x}_2 = \begin{pmatrix} 0.011812 \\ 0.98582 \\ 0.97329 \end{pmatrix}$.
So, the process converges to attractor (0,1,1).

3.4 Proving Convergence of the Autoassociative Recall Algorithm

Lemma 4 *Suppose we have an autoassociative memory with kernel* K, *stored vectors* $\mathbf{x}_1 \ldots \mathbf{x}_m$ *forming columns of matrix* $\mathbf{X}$, *and a matrix* $\mathbf{S}$ *with elements:*

$$s_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$$

The dynamical system corresponding to the associative recall is:

$$\begin{array}{rcl} z_t^{(i)} & = & K(\mathbf{x}_i, \mathbf{x}_t) \\ \mathbf{y}_t & = & \mathbf{X}^T \mathbf{S}^{-1} \mathbf{z} \\ \mathbf{x}_{t+1} & = & f(\mathbf{y}_t) \end{array} \qquad (29)$$

Suppose kernel $K(\mathbf{u},\mathbf{v})$ *is continuous, and it satisfies the piece-wise Mercer condition for a certain* $\delta > 0$ *such that the stored vectors* $\mathbf{x}_1\ \mathbf{x}_2 \ldots\ \mathbf{x}_m$ *satisfy* $\|\mathbf{x}_i - \mathbf{x}_j\| > \delta$ *for* $i \neq j$. *Then attractors of the dynamical system (29) are either fixed points or 2-cycles.*

Proof:

We will prove that $\lim_{t\to\infty}(\mathbf{x}_{t+1} - \mathbf{x}_{t-1}) = 0$. For this we construct an energy function in a way that is analogous to Hopfield-like networks:

$$E_t = -\frac{1}{2} K(\mathbf{x}_t, \mathbf{y}_{t-1}) \qquad (30)$$

Step 1. Show that the energy has lower bound. Because f is bounded, the closure of the co domain of f in (29) is a certain compact set Q. So, $\mathbf{x}_t$ will remain in Q for all $t \geq 1$. The energy (30) is bounded over $Q \times Q$ as a continuous function over a compact set.

Step 2. Show that there exists a projective self-conjugated operator $\mathbf{C}: E_\varphi \to E_\varphi$ such that $\mathbf{C}\varphi(\mathbf{x}_i) = \varphi(\mathbf{x}_i)$, $i = 1 \ldots m$. By theorem 1 there exists a feature space $H \equiv E_\varphi$ such that the kernel gives a scalar product in this space.

For every finite set of vectors $\varphi(\mathbf{x}_i), i = 1 \ldots m$ in H we can construct a projective operator $\mathbf{C}$ that projects to the subspace spanned with $\varphi(\mathbf{x}_i)$. Indeed, applying Gram-Schmidt **orthogonalization** to $\varphi(\mathbf{x}_i)$ (see, e.g. [55]) we can build an orthonormal set (basis) of vectors $\mathbf{x}_i, i = 1 \ldots m$.

Then define an operator as follows:

$$\mathbf{C}(\mathbf{u}) = \sum_{k=1}^{m} < \mathbf{x}_k, \mathbf{u} > \mathbf{x}_k$$

This $\mathbf{C}$ is projective, and it projects to the finite-dimensional subspace $<\xi_i> = <\varphi(\mathbf{x}_i)>$. (Here by $<\xi_i>$ we denote a subspace spanned with all $\mathbf{x}_i, i = 1 \ldots m$).

Step 3. Show that energy decreases monotonically every 2 steps.

Applying properties of the scalar product in E_{φ_x} and symmetry of $\mathbf{C}$ we get:

$$\begin{aligned} E_t - E_{t+1} &= \frac{1}{2}\left(K(\mathbf{y}_t, \mathbf{x}_{t+1}) - K(\mathbf{y}_{t-1}, \mathbf{x}_t)\right) = \\ &= \frac{1}{2}\left(\langle \varphi(\mathbf{y}_t), \varphi(\mathbf{x}_{t+1})\rangle - \langle \varphi(\mathbf{y}_{t-1}), \varphi(\mathbf{x}_t)\rangle\right) = \\ &= \frac{1}{2}\left(\langle \varphi(\mathbf{x}_t), \varphi(\mathbf{x}_{t+1})\rangle - \langle \mathbf{C}\varphi(\mathbf{x}_{t-1}), \varphi(\mathbf{x}_t)\rangle\right) = \\ &= \frac{1}{2}\left(\langle \varphi(\mathbf{x}_t), \varphi(\mathbf{x}_{t+1})\rangle - \langle \varphi(\mathbf{x}_{t-1}), \mathbf{C}\varphi(\mathbf{x}_t)\rangle\right) \\ &= \frac{1}{2}\left(\langle \varphi(\mathbf{y}_t), \varphi(\mathbf{x}_{t+1})\rangle - \langle \varphi(\mathbf{x}_{t-1}), \varphi(\mathbf{y}_t)\rangle\right) = \\ &= \frac{1}{2}\left(K(\mathbf{y}_t, \mathbf{x}_{t+1}) - K(\mathbf{y}_t, \mathbf{x}_{t-1})\right) \geq 0 \end{aligned} \qquad (31)$$

Because $\mathbf{x}_{t+1} = f(\mathbf{y}_t)$, $\mathbf{x}_{t+1}$ is closer to $\mathbf{y}_t$ than any other point on the trajectory, and the kernel is monotonic with respect to distance between $\mathbf{x}$ and $\mathbf{y}$, the expression (31) will be non-negative. Moreover, is $E_t - E_{t+1} \geq 0$; it is zero if and only if a fixed point is reached.

Step 4. Show that the total amount of energy decrease is finite if and only if sequences $\mathbf{x}_{2k}$ **and** $\mathbf{x}_{2k+1}$ **converge.** Suppose the energy lower bound is $-E_*$.

$$\begin{aligned} E_t - E_{t+1} &= \frac{1}{2}\left(<\varphi(\mathbf{x_{t+1}}), \varphi(\mathbf{y_t})> - <\varphi(\mathbf{x_{t-1}}), \varphi(\mathbf{y_t})>\right) \\ &= \frac{1}{2} <\varphi(\mathbf{x_{t+1}}) - \varphi(\mathbf{x_{t-1}}), \varphi(\mathbf{y_t})> = \\ &= \|\varphi(\mathbf{x_{t+1}}) - \varphi(\mathbf{x_{t-1}})\| \cdot \|\varphi(\mathbf{y_t})\| \cdot \cos\theta \end{aligned} \qquad (32)$$

So, there exists $\mu > 0$ such that $E_t - E_{t+1} \geq \mu \|\varphi(\mathbf{x}_{t+1}) - \varphi(\mathbf{x}_{t-1})\|$.

Then $\sum_{t=t_0}^{\infty} \|\mathbf{x}'_{t+1} - \mathbf{x}'_{t-1}\| \leq \sum_{t=t_0}^{\infty} \frac{1}{\mu}(E_t - E_{t+1}) = \frac{1}{\mu}(E_* + E_{t_0})$. The sum on the left hand of this equation will be finite if and only if sequences $\mathbf{x}_{2k}$ and $\mathbf{x}_{2k+1}$ converge, and we are done.

Remark 8 *By Theorem 3 and Remark 2 we have proven that (all) stored patterns are attractors. This means that* $E_t = \mathrm{const}$ *if* $\mathbf{x}$ *equals one of stored vectors.*

We next prove Theorem 2.

Proof. The proof is analogous to the proof of lemma 4. Note that if the k-th attractor is reached $z_k\ z_i = 0, i \neq k$. Activation function in the form of generalized sigmoid brings $\mathbf{z}$ closer to an attractor. So, where in the proof of lemma 4 the linear activation function is replaced with a generalized sigmoid, convergence to an attractor can only be fasted reached.

3.5 Variable Kernel and Dimensionality

For polynomial kernel of degree d we have:

$$\begin{gathered} K(\mathbf{x}, \mathbf{y}) = \frac{1}{3}\left((1 + <\mathbf{x}_a, \mathbf{y}_a>)^d + (1 + <\mathbf{x}_b, \mathbf{y}_b>)^d + \right. \\ \left. + ((1 + <\mathbf{x}_a, \mathbf{y}_a>)(1 + <\mathbf{x}_b, \mathbf{y}_b>))^{d'}\right), \\ d' = \lfloor d/2 \rfloor \end{gathered} \qquad (33)$$

For such decomposed kernels, the matrix $\mathbf{S}^{-1}$ can be efficiently ($\mathrm{O}(m^2)$) updated using formula (16).

Lemma 5 *If the kernel* K *is a linear combination of basis functions, there exists an additive kernel* K_1 *having the same feature space* $E_{\varphi(x)}$.

Having the same feature space is important because it leads to identical behavior of two kernel memories with these to kernels. We note that as before, if $\mathbf{S}$ is an approximately diagonal matrix, the inverse $\mathbf{S}^{-1}$ can be estimated efficiently. A diagonal matrix appears for example in the Radial-Basis-Function-like kernels. In this situation the approximations are given by $\mathbf{S}=\lambda\mathbf{I}+\eta\mathbf{S}_1$.

3.5.1 Memory Stability with Changing Kernels. Suppose all vectors submitted for learning and recall belong to a certain compact set Q. Let us define the norm for kernels:

$$\|K\|^2=\int\int_{Q\times Q} K^2(\mathbf{u},\mathbf{v})d\mathbf{u}d\mathbf{v}$$

Proposition 2 *For given space E_{n+q} and a compact set Q in it there exists a constant $M>0$ such that if $\|K_{n+q}-K_n\|\le M\|K_n\|$ for any set of vectors $\mathbf{x}_1\dots\mathbf{x}_m$ stored in the memory with K_n these vectors remain in attraction basins of corresponding attractors for the memory with kernel K_{n+q} whose attractors expand $\mathbf{x}_1\dots\mathbf{x}_m$ to dimension $n+q$.*

Proof. The proof directly follows from the norm definition and direct estimation of attractor shift when the kernel is changed.

3.6 Proof of Theorem 3

Lemma 6 *Let $f:\mathbb{R}\to\mathbb{R}$ be a generalized sigmoid. Suppose $t\in[0;1]$ and variable $\mathcal{N}$ takes two values 0 and 1. Then there exist constants c and θ in $(0;1)$ such that if $|t-\mathcal{N}|<c$, $|f(t)-\mathcal{N}|\le\theta|t-\mathcal{N}|$*

Proof. By definition 1 there exist c and θ in (0,1) such that $|f'(s)|<\theta$ if $|s-\mathcal{N}|<c$. Estimate:

$$|f(t)-f(\mathcal{N})|=\left|\int_{\mathcal{N}}^{t} f'(s)ds|\le\theta|t-\mathcal{N}\right|$$

Here we prove Theorem 3.

Proof. Select one of stored patterns $\mathbf{x}_{i_0}$ and denote it as $\mathbf{x}_0\equiv\mathbf{x}_{i_0}$. Consider the vector $\zeta=\mathbf{S}^{-1}\mathbf{z}$. If $\mathbf{x}$ is equal to, one of stored vectors, corresponding vector $\zeta_0=\mathbf{S}^{-1}\mathbf{z}_0$ has all zero coordinates except $\zeta_{i_0}^{(0)}=1$, equivalently $\zeta_i^{(0)}=\delta_{i,i_0}$.

Estimate the norm $\|\zeta-\zeta_0\|_\infty$.

$$\|\zeta-\zeta_0\|_\infty\le\|\zeta-\zeta_0\|\le\|\mathbf{S}^{-1}\|\cdot\|\mathbf{z}-\mathbf{z}_0\|\le\|\mathbf{S}^{-1}\|\cdot L\cdot\|\mathbf{x}-\mathbf{x}_0\|$$

If $\|\zeta-\zeta_0\|_\infty<c$, $\|f(\zeta)-f(\zeta_0)\|<\theta\|\zeta-\zeta_0\|_\infty$ according to lemma 6. So, if

$$\|\mathbf{x}-\mathbf{x}_0\|\le\frac{c}{\|\mathbf{S}^{-1}\|\cdot L}$$

iterations of the recall will make ζ converge to ζ_0 with exponential velocity, that immediately implies convergence of $\mathbf{x}_t$ to $\mathbf{x}_0$, and we are done.

3.7 Reconsolidation and Riemannian Distance

3.7.1 Riemannian Metric. Riemannian manifold is a smooth, at least twice differentiable manifold (see [52,56], or any textbook on Riemannian Geometry), which has a scalar product at every point. The scalar product, or Riemannian Metric, is defined in a tangent space of a point as a positive quadratic form.

Riemannian metrics enables to measure curve length and introduce geodesics: trajectories of a free particle attached to the manifold. Between every two points there exist at least one geodesic that have minimal length among all curves joining these two points. Length of this geodesic gives Riemannian distance over the manifold.

There is following Riemannian distance between two kernel associative memories:

$$\rho(X,Y)^2=2(m-tr(\mathbf{S}^{-1/2}\mathbf{Q}_{xy}\mathbf{T}^{-1/2})) \tag{34}$$

Here X and Y are two memories with the same m,n, and K. They have S-matrices $\mathbf{S}$ and $\mathbf{T}$ respectively and $\mathbf{Q}$ is the "cross-matrix": $q_{ij}=K(\mathbf{x}_i,\mathbf{y}_j)$, $i,j=1\dots m$.

3.7.2 Geodesic Update. To find a point analogous to convex combination of $\mathbf{X}_0$ and $\mathbf{X}_1$ we build a geodesic $\gamma_{X_0}^{X_1}$ joining these to points, and take a point $X_1^*=\gamma_{X_0}^{X_1}(\tau)$. Here $\tau\in[0,1]$ is a step parameter related to size of a shift during each update. For $t=0$ we stay at $\mathbf{X}_0$, for $t=1$ the point is changed to $\mathbf{X}_1$. Repeatedly, we track from $\mathbf{X}_1$ to $\mathbf{X}_2$ when a stimulus $x_{1,2}$ appears, etc.

The algorithm of memory update using geodesics uses the property of kernel memory that an arbitrary attractor can be added or removed to the network in $O(m^2+n)$ operations with no impact to all other attractors.

Acknowledgments

We express our appreciation to Eric Goldstein for helping to edit this paper as well as to Yariv Levy, Megan Olsen, David Cooper, and Kun Tu for helpful discussions and their continuous support.

Author Contributions

Conceived and designed the experiments: DN HS. Performed the experiments: DN. Wrote the paper: DN HS.

References

1. Amit D (1989) Modeling Brain function. The world of attractor networks Cambridge Univ. Press. 528 p.
2. Leutgeb JK, Leutgeb S, Treves A, Meyer R, Barnes CA, et al. (2005) Progressive transformation of hippocampal neuronal representations in morphed environments. Neuron 48: 345–358.
3. Chafee MV, Goldman-Rakic PS (1998) Matching patterns of activity in primate prefrontal area 8a parietal area 7ip neurons during a spatial working memory task. J Neurophysiol 1998: 2919–2940.
4. Li Z (2001) Computational design and nonlinear dynamics of a recurrent network model of the primary visual cortex. Neural Computation 13: 1749–1780.
5. Seung HS (1998) Continuous attractors and oculomotor control. Neural Networks 11: 1253–1258.
6. Wang XJ (2001) Synaptic reverberation underlying mnemonic persistent activity. Trends Neuroscience 24: 455–463.
7. Fuster JM, Alexande G (1971) Neuron activity related to short-term memory. Neuron 14: 477–485.
8. Miyashita Y (1988) Neuronal correlate of visual associative longterm memory in the primate temporal cortex. Nature 335: 817–820.
9. Miyashita Y, Chang HS (1988) Neuronal correlate of pictorial short-term memory in the primate temporal cortex. Nature 331: 68–70.
10. Kay LM, Lancaster LR, Freeman WJ (1996) Reafference and attractors in the olfactory system during odor recognition. Int J Neural Systems 7: 489–495.
11. Ericson C, Desimone R (1999) Responses of macaque perirhinal neurons during and after visual stimulus association learning. J Neurosci 19: 10404–10416.
12. Compte A, Brunel N, Goldman-Rakic PS, Wang XJ (2000) Synaptic mechanisms and network dynamics underlying spatial working memory in a cortical network model. Cerebral Cortex 10: 910–923.
13. Wills TJ, Lever C, Cacucci F, Burgess N, O'Keefe J (2005) Attractor dynamics in the hippocampal representation of the local environment. Science 308: 873–876.
14. Hopfield JJ (1982) Neural networks and physical systems with emergent collective computational abilities. Proc Natl Acad Sci USA 79: 2554–2558.
15. McNaughton BL, Morris RGM (1987) Hippocampal synaptic enhancement and information storage within a distributed memory system. Trends Neurosci 10: 408–415.

16. Treves A, Rolls ET (1992) Computational constraints suggest the need for two distinct input systems to the hippocampal ca3 network. Hippocampus 2: 189–199.
17. Hasselmo ME, Schnell E, Barkai E (1995) Dynamics of learning and recall at excitatory recurrent synapses and cholinergic modulation in rat hippocampal region ca3. J Neurosci 15: 5249–5262.
18. Durstewitz D, Seamans JK, Sejnowski TJ (2000) Dopamine mediated stabilization of delay-period activity in a network model of prefrontal cortex. J Neurophysiol 83: 1733–1750.
19. Blumenfeld B, Preminger S, Sagi D, Tsodyks M (2006) Dynamics of memory representations in networks with novelty-facilitated synaptic plasticity. Neuron 52: 383–394.
20. Gruber AJ, Dayan P, Gutkin BS, Solla SA (2006) Dopamine modulation in the basal ganglia locks the gate to working memory. J Comput Neurosci 20: 153–166.
21. Treves A (1990) Graded-response neurons and information encodings in autoassociative memories. Phys Rev A 42: 2418–2430.
22. Lengyel M, Kwag J, Paulsen O, Dayan P (2005) Matching storage and recall: hippocampal spike–timing–dependent plasticity and phase response curves. Nature Neuroscience 8: 1677–1683.
23. Poggio T, Girosi F (1990) Regularization algorithms for learning that are equivalent to multilayer networks. Science 247: 978–982.
24. Billings SA, Fung CF (1995) Recurrent radial basis function networks for adaptive noise cancellation. Neural Networks 8: 273–290.
25. Cheung YM (2002) A new recurrent radial basis function network. Neural Information Processing, ICONIP '02 Proc of the 9th International Conference on 2: 1032–1036.
26. Miyoshi T, Ichihashi H, Okamoto S, Hayakawa T (1995) Learning chaotic dynamics in recurrent rbf network. Neural Networks, Proc IEEE International Conference on. pp 588–593.
27. Sun J, Zhang T, Liu H (2004) Modelling of chaotic systems with novel weighted recurrent least squares support vector machines. Lecture Notes in Computer Science 3173: 578–585.
28. Dudai Y (1997) Time to remember. Neuron 18: 179–182.
29. Eichenbaum H (2006) The secret life of memories. Neuron 50: 350–352.
30. Dudai Y, Eisenberg M (2004) Rite of passage of the engram: Reconsolidation and the lingering consolidation hypothesis. Neuron 44: 93–100.
31. Nader K, Schafe GE, LeDoux JE (2000) Fear memories require protein synthesis in the amygdala for reconsolidation after retrieval. Nature 406: 722–726.
32. Lee JLC, Everitt BJ, Thomas KL (2004) Independent cellular processes for hippocampal memory consolidation and reconsolidation. Science 304: 839–843.
33. Medina JH, Bekinschtein P, Cammarota M, Izquierdo I (2008) Do memories consolidate to persist or do they persist to consolidate? Behavioural Brain Research. pp 61–69.
34. Siegelmann HT (2008) Analog-symbolic memory that tracks via reconsolidation. Physica D 237: 1207–1214.
35. Preminger S, Sagi D, Tsodyks M (2005) Morphing visual memories through gradual associations. Perception Suppl 34: 14.
36. McClelland JL, Rumelhart DE (1981) An interactive activation model of context effects in letter perception: Part i. an account of basic findings. Psychological Review 88: 375–407.
37. Vapnik V (1998) Statistical Learning Theory. NY: John Wiley & Sons. 736 p.
38. Cucker F, Smale S (2002) On the mathematical foundations of learning. Bulletin of American Mathematical Society 39: 1–49.
39. Ben-Hur A, Horn D, Siegelmann HT, Vapnik V (2001) Support vector clustering. Journal of Machine Learning Research 2: 125–137.
40. Zhang BL, Zhang H, Ge SS (2004) Face recognition by applying wavelet subband representation and kernel associative memory. IEEE Trans Neural Networks 15: 166–177.
41. Caputo B, Niemann H (2002) Storage capacity of kernel associative memories. In: Artificial Neural Networks — ICANN 2002. Heidelberg: Springer Berlin. pp 134–143.
42. Casali D, Constantini G, Perfetti R, Ricci E (2006) Associative memory design using support vector machines. IEEE Trans on Neural Networks 17: 1165–1174.
43. Zemel RS, Mozer MC (2001) Localist attractor networks. Neural Computation 13: 1045–1064.
44. Albert A (1972) Regression and the Moore-Penrose pseudoinverse. NY-London: Academic Press. 180 p.
45. Fusi S, Abbott LF (2007) Limits on the memory storage capacity of bounded synapses. Nature Neuroscience 10: 485–493.
46. Petersen KB, Pedersen MS (2008) The Matrix Cookbook. Available: http://matrixcookbook.com/.
47. LeCun Y, Botton L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86: 2278–2324.
48. Zhang R, Rudnicky AI (2002) A large scale clustering scheme for kernel k-means. 16th International Conference on Pattern Recognition (ICPR'02) 4: 40289.
49. Tronson NC, Taylor JR (2007) Molecular mechanisms of memory reconsolidation. Nature Reviews Neuroscience 8: 262–275.
50. Singer E (2009) Manipulation memory. Technology Review 46: 138–139.
51. Monfils MH, Cowansag KK, Klann E, LeDoux JE (2009) Extinction-reconsolidation boundaries: Key to persistent attenuation of fear memories. Science 324: 951–955.
52. do Carmo MP (1992) Riemannian Geometry. Brikhauser. 300 p.
53. Minear M, Park DC (2004) A lifespan database of adult facial stimuli. Behavior Research Methods, Instruments, & Computers 36: 630–633.
54. Sanderson C (2008) Biometric Person Recognition: Face, Speech and Fusion. VDM-Verlag. 156 p.
55. Golub GH, Van Loan CF (1996) Matrix computations (3rd ed.). Baltimore, MD, USA: Johns Hopkins University Press. 694 p.
56. Petersen P (2006) Riemannian Geometry. Graduate Texts in Mathematics. Springer. 408 p.