JACOBS UNIVERSITY

Xu He, Herbert Jaeger

Overcoming Catastrophic Interference by Conceptors

Technical Report No. 35
May 2017

School of Engineering and Science

Overcoming Catastrophic Interference by Conceptors

Xu He, Herbert Jaeger

School of Engineering and Science
Jacobs University Bremen gGmbH
Campus Ring 12
28759 Bremen
Germany

E-Mail: x.he@jacobs-university.de
http://minds.jacobs-university.de/

Abstract

Catastrophic interference has been a major roadblock in the research of continual learning. Here we propose a variant of the back-propagation algorithm, "conceptor-aided back-prop" (CAB), in which gradients are shielded by conceptors against degradation of previously learned tasks. Conceptors have their origin in reservoir computing, where they have been previously shown to overcome catastrophic forgetting. CAB extends these results to deep feedforward networks. On the disjoint MNIST task CAB outperforms two other methods for coping with catastrophic interference that have recently been proposed in the deep learning field.

Contents

1 Introduction

When trained on a sequence of tasks individually, neural networks usually forget about previous tasks after the weights are adjusted for a new task. This notorious problem known as catastrophic interference [French, 1999; McCloskey and Cohen, 1989; Kumaran et al., 2016; Ratcliff, 1990] poses a serious challenge towards continual learning. An effective solution to this problem using conceptors was proposed by the second author [Jaeger, 2014] in a reservoir computing context for incrementally loading dynamic patterns to a reservoir. Adopting and extending those methods, here we propose a *conceptor-aided back-propagation* (CAB) algorithm to train feed-forward networks, and compare its performance to recent works that address the same problem.

The rest of this report is structured as follows. Section 2 introduces conceptors and their application to incremental learning by linear regression. Section 3 extends the method to stochastic gradient descent and describes the CAB algorithm. Section 4 demonstrates the performance of CAB on the permuted MNIST task and the disjoint MNIST task. Finally in Section 5 we discuss the limitation of this method and its future extension.

2 Incremental Ridge Regression by Conceptors

In this section, we review the basics of conceptor theory and its application to incrementally training linear readouts of recurrent neural networks (RNNs) as used in reservoir computing. A comprehensive treatment can be found in [Jaeger, 2014].

2.1 Conceptors

In brief, a *matrix conceptor* C for some vector-valued random variable $x \in \mathbb{R}^N$ is defined as a linear transformation that minimizes the following loss function.

$$\mathbb{E}_x[||x - Cx||^2] + \alpha^{-2}||C||^2_{\text{fro}} \tag{1}$$

where α is a control parameter called *aperture* and $||\cdot||_{\text{fro}}$ is the Frobenius norm. This optimization problem has a closed-form solution

$$C = R(R + \alpha^{-2}I)^{-1} \tag{2}$$

where $R = \mathbb{E}_x[xx^\top]$ is the $N \times N$ correlation matrix of x, and I is the $N \times N$ identity matrix. This result given in (2) can be understood by studying the singular value decomposition (SVD) of C. It has been shown that if $R = U\Sigma U^\top$ is the SVD of R, then the SVD of C is given as $USU^\top$, where the singular values s_i of C can be written in terms of the singular values σ_i of R: $s_i = \sigma_i/(\sigma_i + \alpha^{-2}) \in [0, 1)$. In intuitive terms, C is a soft projection matrix on the linear subspace where the samples of x lie, such that for a vector y in this subspace, C acts like the identity: $Cy \approx y$, and when some noise ϵ orthogonal to the subspace is added to y, C de-noises: $C(y+\epsilon) \approx y$. We define the quota $Q(C)$ of a conceptor to be the mean singular values: $Q(C) := \frac{1}{N}\sum_{i=1}^{N} s_i$. Intuitively, the quota measures the fraction of the total dimensions of the entire vector space that is claimed by C.

Moreover, logic operations that satisfy many laws of Boolean logic can be defined on matrix conceptors as the following:

$$\neg C := I - C, \tag{3}$$

$$C^i \vee C^j := (R^i + R^j)(R^i + R^j + \alpha^{-2} I)^{-1} \tag{4}$$

$$C^i \wedge C^j := \neg(\neg C^i \vee \neg C^j) \tag{5}$$

2.2 Incremental Ridge Regression

With the help of these logic operations, conceptors can be applied to incrementally train one linear model by ridge regression for multiple input-to-output mapping tasks, such that (i) learning a new task does not interfere with previously learned tasks; (ii) similarities between tasks are exploited; (iii) the amount of remaining memory space can be monitored. Here “memory space” refers to the linear space of input vectors.

In particular, consider a sequence of m incoming tasks indexed by j and we denote the training dataset for the j-th task by $\{(x_1^j, y_1^j), \cdots, (x_n^j, y_n^j)\}$, where $x_i^j \in \mathbb{R}^N$ are input vectors and $y_i^j \in \mathbb{R}^M$ their corresponding target outputs. Whenever the training dataset for a new task is available, the incremental learning method will compute a matrix conceptor C^j for the input variable of the new task and update the linear model, resulting in a sequence of linear models $W^1, \ldots W^m$ such that W^j solves not only the j-th task but also all previous tasks: for $k \leq j$, $y^k \approx W^j x^k$. The conceptor C^j characterizes the memory space occupied by the j-th task, and we use $A^{j-1} = C^1 \vee \cdots \vee C^{j-1}$ and $F^j = \neg A^{j-1}$ to represent the memory space already claimed by the tasks $1, \ldots, j-1$ and the memory space still free for the j-th task, respectively. More specifically, this method proceeds in the following way:

- **Initialization (no task trained yet):** $W^0 = 0_{M \times N}, A^0 = 0_{N \times N}$.
- **Incremental task learning:** For $j = 1, \ldots, m$ do:
  1. Store the input vectors from the j-th training dataset of size n into a $N \times n$ sized input collection matrix X^j, and store the output vectors into a $M \times n$ sized output collection matrix Y^j.
  2. Compute the conceptor for this task by $C^j = R^j(R^j + \alpha^{-2} I)^{-1}$, where $R^j = \frac{1}{n} X^j X^{j\top}$
  3. Train a matrix W_{inc}^j (to be added to W^{j-1}, yielding W^j):

(a) $F^j := \neg A^{j-1}$ (*comment: this conceptor characterizes the "still disposable" memory space for the j-th task*),

(b) $T := Y^j - (W^{j-1}X^j)$ (*comment: this matrix consists of target values for a linear regression to compute W^j_{inc}*),

(c) $S := F^j X^j$ (*comment: this matrix consists of arguments for the linear regression*),

(d) $W^j_{inc} = ((SS^\top/n + \lambda^{-2}I)^{-1}ST^\top/n)^\top$ (*comment: carry out the regression, regularized by λ^{-2}*),

4. Update W^j: $W^j = W^{j-1} + W^j_{inc}$.
5. Update A : $A^j = A^{j-1} \vee C^j$ (*comment: this is possible due to the associativity of the OR operation on conceptors*)

Intuitively speaking, when learning a new task, this algorithm leaves the already trained directions in the memory space (characterized by A^{j-1}) intact and exploits only the components of input vectors in the free space (characterized by F^j) to compensate errors for the new task.

3 Conceptor-Aided SGD and Back-prop

3.1 SGD

In the algorithm introduced in the previous section, W^j_{inc} is computed by ridge regression, which gives the closed-form solution to minimize the cost function

$$\mathcal{J}(W^j_{inc}) := \mathbb{E}[|W^j_{inc}s - t|^2] + \lambda^{-2}|W^j_{inc}|^2_{\text{fro}} \tag{6}$$

where $t = y^j - W^{j-1}x^j, s = F^j x^j$. One can also minimize this cost function by stochastic gradient descent (SGD), which starts from an initial guess of W^j_{inc} and repeatedly performs the following update

$$W^j_{inc} \leftarrow W^j_{inc} - \eta \nabla_{W^j_{inc}} \mathcal{J}(W^j_{inc}) \tag{7}$$

where η is the learning rate and the gradient is given by:

$$\nabla_{W^j_{inc}} \mathcal{J}(W^j_{inc}) = 2\mathbb{E}[(W^j_{inc}s - t)s^\top] + 2\lambda^{-2}W^j_{inc} \tag{8}$$

Substituting $t = y^j - W^{j-1}x^j, s = F^j x^j = (I - A^{j-1})x^j$ in (8), we get

$$\begin{aligned}\nabla_{W^j_{inc}} \mathcal{J}(W^j_{inc}) &= 2\mathbb{E}[(W^j_{inc}(I - A^{j-1})x^j - y^j + W^{j-1}x^j)s^\top] + 2\lambda^{-2}W^j_{inc} \\ &= 2\mathbb{E}[(-W^j_{inc}A^{j-1}x^j + (W^{j-1} + W^j_{inc})x^j - y^j)s^\top] + 2\lambda^{-2}W^j_{inc} \end{aligned} \tag{9}$$

Due to the regularization term in the cost function, as the optimization goes on, eventually W_{inc} will null the input components that are not inside the linear subspace characterized by F^j, hence $W^j_{inc}A^{j-1}x^j$ will converge to 0 as the algorithm proceeds. In addition, since $W^j = W^{j-1} + W^j_{inc}$, (9) can be simplified to

$$\nabla_{W^j_{inc}} \mathcal{J}(W^j_{inc}) = 2\mathbb{E}[(W^j x^j - y^j)s^\top] + 2\lambda^{-2}W^j_{inc} \tag{10}$$

Adding W^{j-1} to both sides of (7), we obtain a update rule for W^j:

$$W^j \leftarrow W^j - 2\eta\mathbb{E}[es^\top] + 2\eta\lambda^{-2}W^j_{inc} \tag{11}$$

where $e := W^j x^j - y^j$. In practice, at every iteration, the expected value can be approximated by a mini-batch of size n_B, indexed by i_B:

$$\hat{\mathbb{E}}[es^\top] = \frac{1}{n_B}\sum_{i_B=0}^{L}(W^j x^j_{i_B} - y^j_{i_B})(F^j x^j_{i_B})^\top = \frac{1}{n_B}\sum_{i_B=0}^{L}(W^j x^j_{i_B} - y^j_{i_B})x^{j\top}_{i_B}F^j \tag{12}$$

where the transpose for F^j can be dropped since it is symmetric.

If we only train the $j-$th task without considering the previous tasks, the update rule given by normal SGD will be

$$W^j \leftarrow W^j - 2\eta\mathbb{E}[ex^{j\top}] + 2\eta\alpha^{-2}W^j \tag{13}$$

Comparing this to the update rule in (11), we notice two modifications when a conceptor is adopted to avoid forgetting: first, the conceptor-projected input vector $s = F^j x^j$ instead of the original input vector x^j is used to calculate the gradient of weights; second, regularization is done on the weight increment W^j_{inc} rather than the final weight W^j. These two modifications lead to our design of a conceptor-aided algorithm for training multilayer feed-forward networks.

3.2 Backprop

The basic idea of CAB is to guide the gradients of the loss function on every linear component of the network by a matrix conceptor computed from previous tasks during error back-propagation, repeatedly applying the conceptor-aided SGD technique introduced in the previous section.

Consider a feed-forward network with $L+1$ layers, indexed by $l = 0, \dots L$, such that the 0-th and the L-th layers are the input and output layers respectively. $W^{(l)}$ represents the linear connections between the $(l-1)$-th and the l-th layer, where we refer to the former as the pre-synaptic layer with respect to $W^{(l)}$, and to the latter as the post-synaptic layer. We denote by $N^{(l)}$ the size of the l-th layer (excluding the bias unit) and $A^{(l)^j}$ a conceptor characterizing the memory space in the l-th layer used up by the first j tasks. Let $\sigma(\cdot)$ be the activation function of the nonlinear neurons and θ all the parameters of the network to be trained. Then the incremental training method with CAB proceeds as follows:

- **Initialization (no task trained yet):** $\forall l = 0, \ldots, L{-}1$, ${A^{(l)}}^0 := 0_{(N^{(l)}+1)\times(N^{(l)}+1)}$, and randomly initialize ${W^{(l+1)}}^0$ to be a matrix of size $N^{(l+1)} \times (N^{(l)} + 1)$.
- **Incremental task learning:** For $j = 1, \ldots, m$ do:

  1. $\forall l = 0, \ldots, L - 1, {F^{(l)}}^j = \neg {A^{(l)}}^{(j-1)}$. (*This conceptor characterizes the still disposable memory space in layer l for learning task j*)
  2. Update the network parameters from $\theta^{(j-1)}$ to θ^j by stochastic gradient descent, where the gradients are computed by CAB instead of the classical backprop. Algorithms 1 and 2 detail the forward and backward pass of CAB, respectively. Different from classical backprop, the gradients are guided by a matrix conceptor ${F^{(l)}}^j$, such that in each layer only the activity in the still disposable memory space will contribute to the gradient. Note that the conceptors remain the same until convergence of the network for task j.
  3. After training on the j-th task, run the forward procedure again on a batch of n_B input vectors, indexed by i_B, taken from the j-th training dataset, to collect activations ${h_{i_B}^{(l)}}^j$ of each layer into a $N^{(l)} \times n_B$ sized matrix ${H^{(l)}}^j$, and set the correlation matrix ${R^{(l)}}^j = \frac{1}{n_B}{H^{(l)}}^j({H^{(l)}}^j)^\top$.
  4. Compute a conceptor on the l-th layer for the j-th pattern by ${C^{(l)}}^j = {R^{(l)}}^j({R^{(l)}}^j + \alpha^{-2}I_{N^{(l)}\times N^{(l)}})^{-1}, \forall l = 0, \ldots, L - 1$. The aperture is set by trial and error, preferably in a cross-validation scheme.
  5. Update the conceptor for already used spaces in every layer: ${A^{(l)}}^j = {A^{(l)}}^j \vee {C^{(l)}}^j, \forall l = 0, \ldots, L - 1$.

4 Experiments

4.1 Disjoint MNIST Experiment

To test the performance of CAB, we applied it on the task of 10-class categorization for disjoint MNIST datasets [Srivastava et al., 2013b; Lee et al., 2017], where the original MNIST dataset is divided into two disjoint datasets with the first one consisting of data for the first five digits (0 to 4), and the second one of the remaining five digits (5 to 9). This task requires a network to learn these two datasets one after the other, then examines its performance of classifying the entire MNIST testing dataset into 10 classes. The current state-of-the-art accuracy on this task, averaged over 10 learning trials, is 94.12(±0.27), achieved by a method called Incremental Moment Matching (IMM) introduced in [Lee et al., 2017]. They also reported the performance of Elastic Weight Consolidation (EWC) method proposed in [Kirkpatrick et al., 2017] to be 52.72(±1.36) on this task.

Algorithm 1 The forward procedure of conceptor-aided backprop for the j-th task, adapted from [Goodfellow et al., 2016]. Input vectors are passed through a feed-forward network to compute the cost function. $\mathcal{L}(\hat{y}^j, y^j)$ denotes the loss for the j-th task, to which a regularizer $\Omega(\theta^j_{inc}) = \Omega(\theta^j - \theta^{j-1}) = ||\theta^j - \theta^{j-1}||^2_{\text{fro}}$ is added to obtain the total cost $\mathcal{J}$, where θ contains all the weights (biases are considered as weights connected to the bias units). The update of parameters rather than the parameters themselves are regularized, similar to the conceptor-aided SGD.

```
Require: Network depth, $l$
Require: $W^{(l)^j}, l \in \{1, \dots, L\}$, the weight matrices of the network
Require: $x^j$, one input vector of the $j$-th task
Require: $y^j$, the target output for $x^j$
$h^{(0)} = x^j$
for $l = 1, \dots L$ do
    $b^{(l)} = [h^{(l-1)\top}, 1]^\top$, include the bias unit
    $a^{(l)} = W^{(l)^j} b^{(l)}$
    $h^{(l)} = \sigma(a^{(l)})$
end for
$\hat{y}^j = h^{(l)}$
$\mathcal{J} = \mathcal{L}(\hat{y}^j, y^j) + \lambda\Omega(\theta^j_{inc})$
```

For this task, we trained a feed-forward network with [784-800-10] neurons. Logistic sigmoid neurons are used in both hidden and output layers, and the network is trained with vanilla SGD to minimize mean squared error. An aperture $\alpha = 9$ was used for all conceptors on all layers, learning rate η and regularization coefficient λ were chosen to be 0.1 and 0.005 respectively. The accuracy of CAB on this task, measured by repeating the experiment 10 times, is $94.91(\pm 0.30)$. It is important to mention that the networks used in [Lee et al., 2017] for IMM and EWC had [784-800-800-10] neurons and rectified linear units (ReLU), so CAB achieved the state-of-the-art performance with less layers and neurons.

4.2 Permuted MNIST Experiment

Another task on which we tested CAB is the permuted MNIST experiment [Goodfellow et al., 2014; Kirkpatrick et al., 2017; Lee et al., 2017; Srivastava et al., 2013a], where a sequence of pattern recognition tasks are created from the MNIST dataset [LeCun et al., 1998]. For each task, a random permutation of input image pixels is generated and applied to all images in MNIST to obtain a new shuffled dataset, equally difficult to recognize as the original one, the objective of each task is to recognize these images with shuffled pixels.

For a proof-of-concept demonstration, we trained a simple but sufficient feed-forward network with [784-100-10] of neurons to classify 10 permuted MNIST datasets. Figure 1 shows the performance of CAB on this task, the average testing

Algorithm 2 The backward procedure of conceptor-aided backprop for the j-th task, adapted from [Goodfellow et al., 2016]. The gradient g of the loss function $\mathcal{L}$ on the activations $a^{(l)}$ represents the error for the linear transformation ${W^{(l)}}^j$ between the $(l-1)$-th and the $l-$th layers. In the standard backprop algorithm, the gradient of $\mathcal{L}$ on ${W^{(l)}}^j$ is computed as an outer product of the post-synaptic errors g and the pre-synaptic activities $h^{(l-1)}$. This resembles the computation of the gradient in the linear SGD algorithm, which motivates us to apply conceptors in a similar fashion as in the conceptor-aided SGD. Specifically, we project the gradient $\nabla_{{W^{(l)}}^j}\mathcal{L}$ by the matrix conceptor ${F^{(l-1)}}^j$ that indicates the free memory space on the pre-synaptic layer.

```
$$g \leftarrow \nabla_{\hat{y}}\mathcal{J} = \nabla_{\hat{y}}\mathcal{L}(\hat{y}, y)$$
**for** $l = L, L-1, \ldots, 1$ **do**
    Convert the gradient on the layer's output into a gradient on the pre-nonlinearity activation ($\odot$ denotes element-wise multiplication):
    $$g \leftarrow \nabla_{a^{(l)}}\mathcal{J} = g \odot \sigma'(a^{(l)})$$
    Compute gradients on weights, projected by ${F^{(l-1)}}^j$, and added to the regularization term on the increment:
    $$\begin{aligned}\nabla_{{W^{(l)}}^j}\mathcal{J} =& g({F^{(l-1)}}^j b^{(l-1)})^\top + \lambda\nabla_{{W^{(l)}}^j}\Omega(\theta^j_{inc}) = g{b^{(l-1)}}^\top {F^{(l-1)}}^j + 2\lambda {W^{(l)}_{inc}}^j \\ =& g{b^{(l-1)}}^\top {F^{(l-1)}}^j + 2\lambda({W^{(l)}}^j - {W^{(l)}}^{j-1})\end{aligned}$$
    Propagate the gradients w.r.t. the next lower-level hidden layers activations:
    $$g \leftarrow \nabla_{h^{(l-1)}}\mathcal{J} = {{W^{(l)}}^j}^\top g$$
**end for**
```

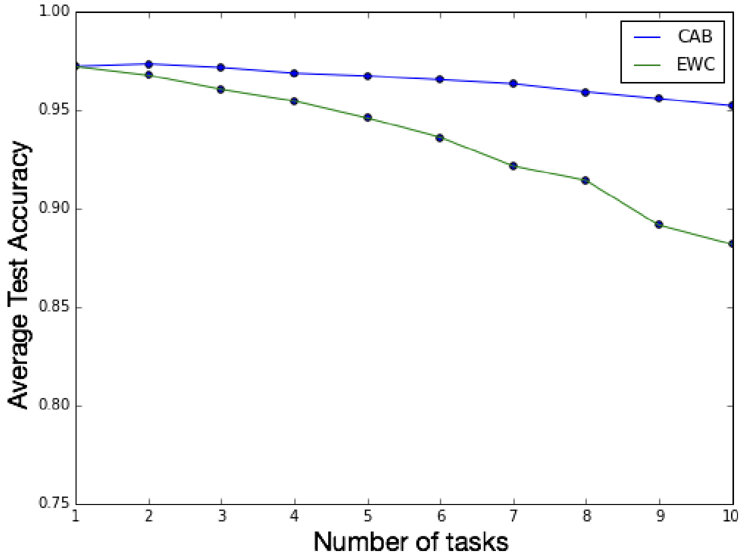
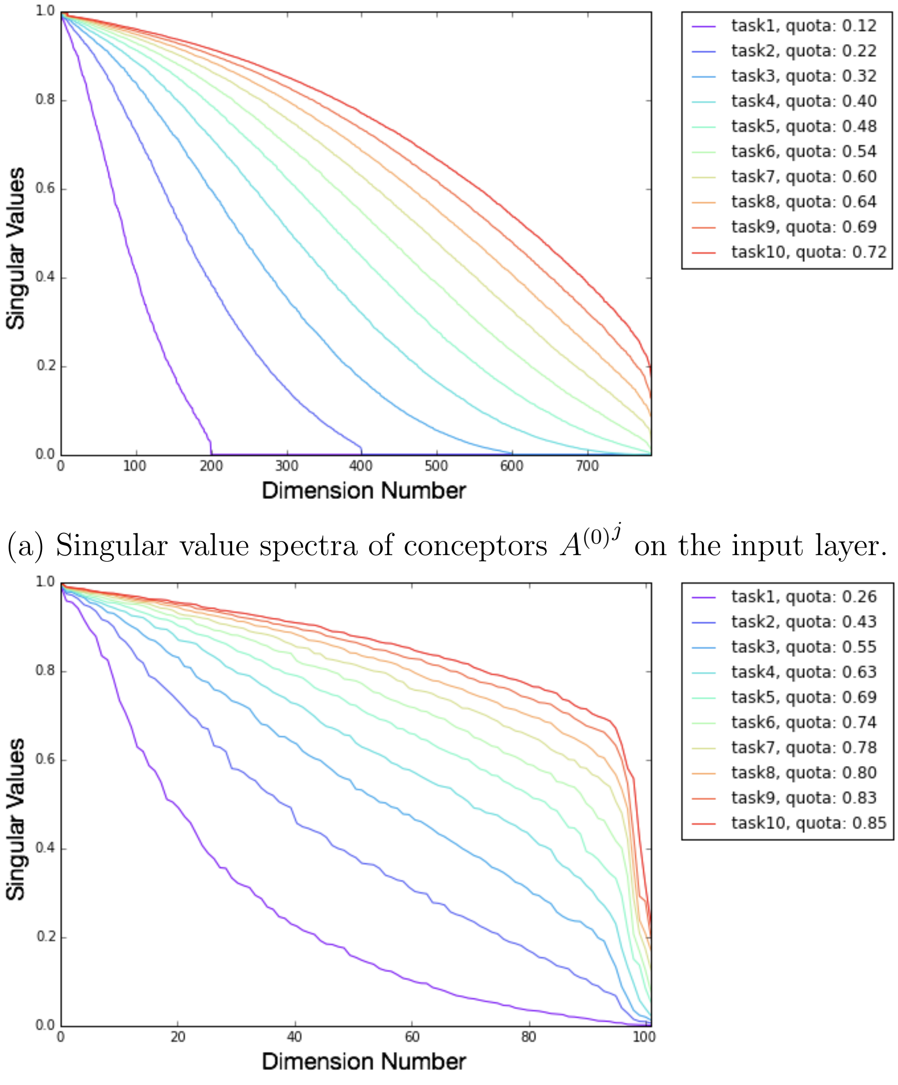


Figure 1: Average performance across different number of permuted MNIST tasks using CAB or EWC

accuracy is 95.2 after learning all 10 tasks sequentially. The network has logistic sigmoid neurons in both hidden and output layers, and is trained with mean squared error as the cost function. Vanilla SGD was used in all experiments to optimize the cost function. Learning rate and aperture were set to 0.1 and 4, respectively. For comparison, we also tested EWC on the same task with the same network architecture, based on the implementation from [Seff, 2017]. The parameters chosen for the EWC algorithm were 0.01 for the learning rate and 15 for the weight of the Fisher penalty term. Although a fair amount of effort was spent on optimizing the parameters of EWC, the accuracies shown here might still be far from its best performance.

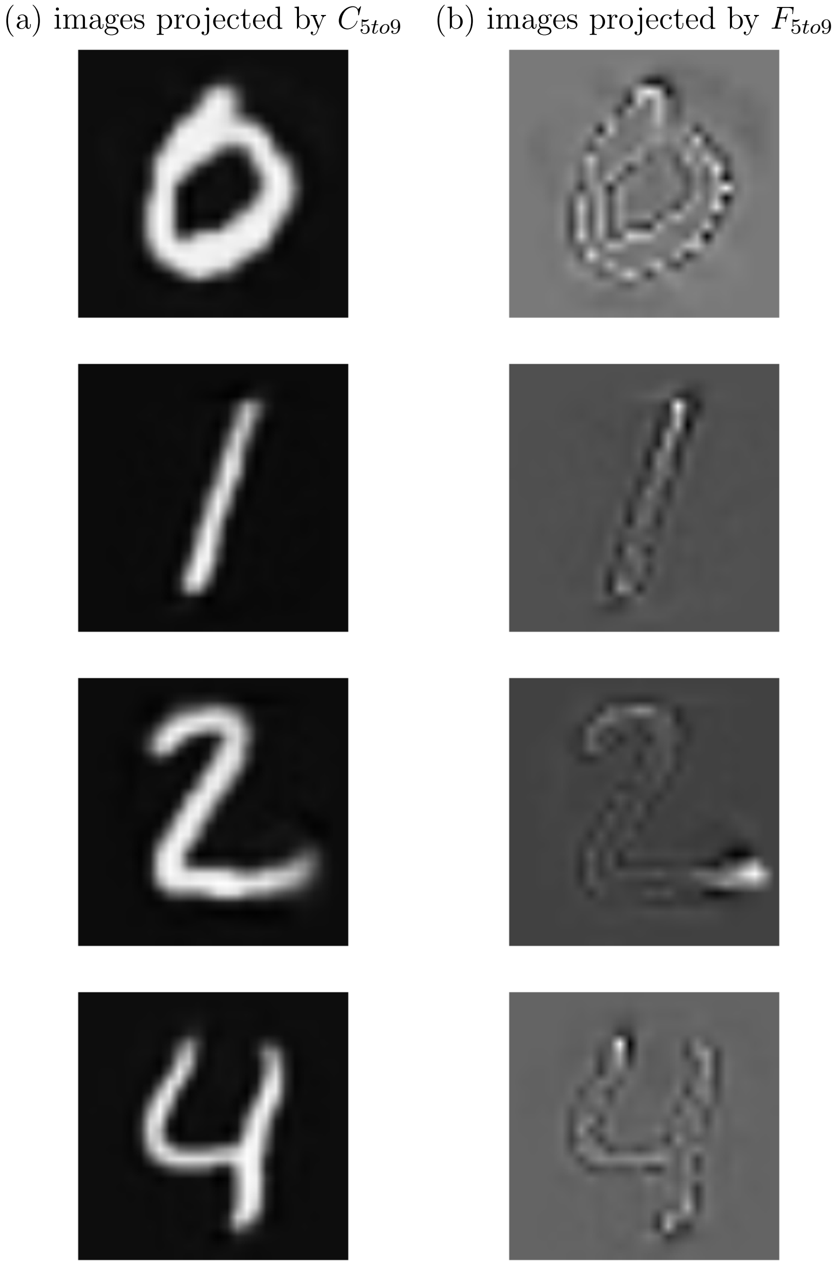
Since all tasks are generated by permuting the same dataset, they should occupy the same portion of the input space. However, as more tasks are learned, the chance that the space of a new task will overlap with the already used input space increases. This can be seen clearly from Figure 2, which shows the singular value spectra and the quota of conceptors for the already used spaces on the input and hidden layers respectively. As the incremental learning proceeds, it becomes less likely for a new task to be in the free space: the second task increases the quota of the input layer memory space by 0.1, whereas the 10th task increases it by only 0.03. However, CAB still manages to make the network learn new tasks based on their components in the non-overlapping space.



(a) Singular value spectra of conceptors $A^{(0)^j}$ on the input layer.

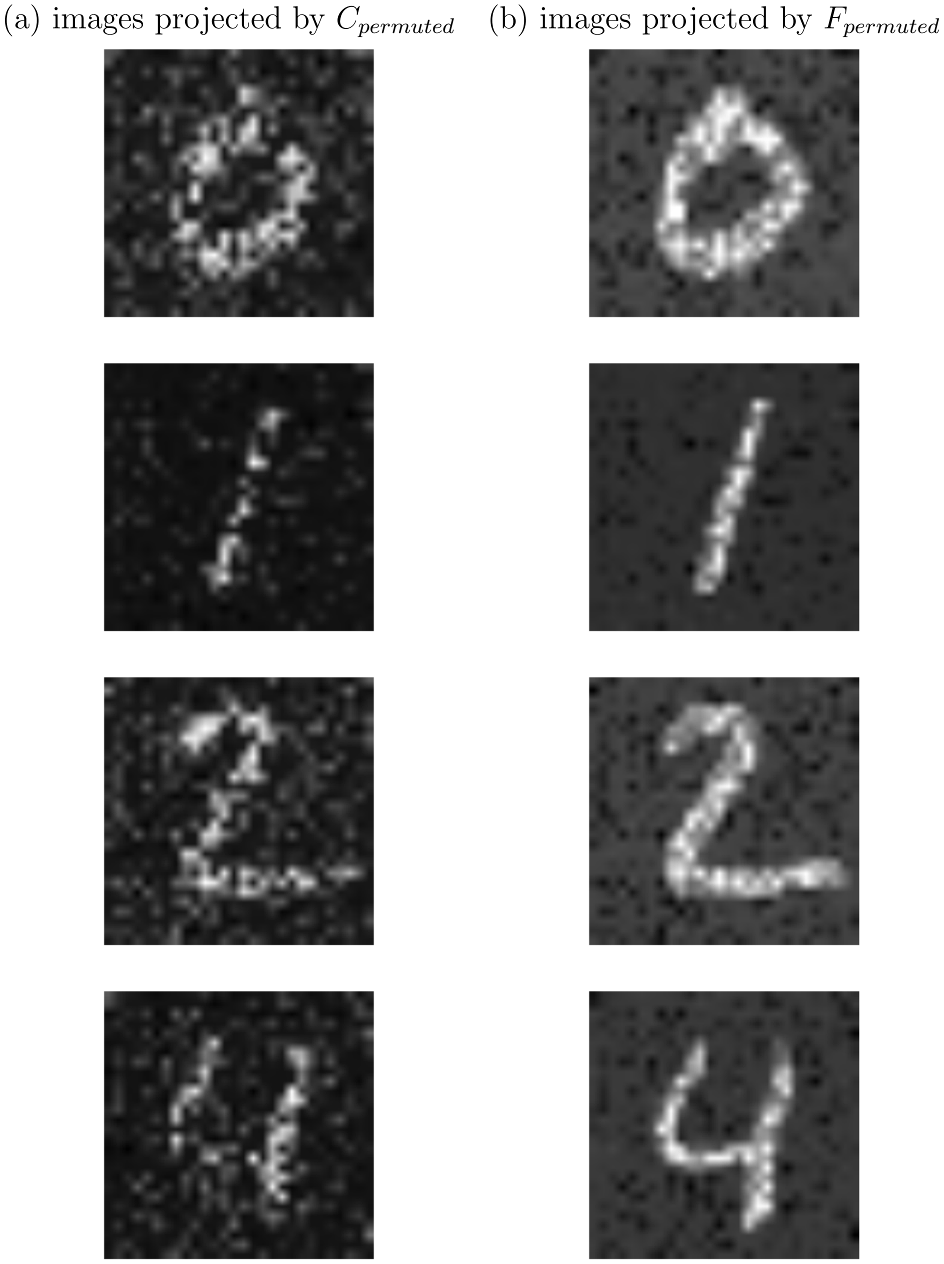
(b) Singular value spectra of conceptors $A^{(1)^j}$ on the hidden layer.

Figure 2: The development of singular value spectra of conceptors for "used-up" space on the input layer and hidden layer during incremental learning of 10 permuted MNIST tasks. Quota of these conceptors are displayed in the legends.



(a) images projected by C_{5to9} (b) images projected by F_{5to9}

Figure 3: Projecting several images from the dataset for digits 0 to 4 by the conceptor C_{5to9} and its negation F_{5to9}. After a network is trained on digits 5 to 9, CAB will only use the components projected by F_{5to9} to correct the classification errors on these images.



(a) images projected by $C_{permuted}$ (b) images projected by $F_{permuted}$

Figure 4: Projecting images from the original MNIST dataset by the conceptor $C_{permuted}$ computed from the shuffled MNIST dataset and its negation $F_{permuted}$. After a network is trained on the shuffled MNIST, CAB will only use the components projected by $F_{permuted}$ to correct the classification errors on these images.

5 Discussion

The experiment results from the previous section indicate that learning disjoint MNIST datasets is a more challenging task for CAB than learning the permuted MNIST datasets. To see why this is the case, it is important to understand that when learning a new task, CAB exploits only the components of input vectors that are not inside the linear subspace characterized by the conceptor of previous tasks. In other words, the more overlap there is between the conceptor of the new task and that of the already learned tasks, the less components in the input patterns are left for CAB to correct the network's errors on the new task, hence more difficult for the network to learn these tasks sequentially.

To quantify the overlap between two conceptors C^i, C^j, we can use the similarity measure proposed in [Jaeger, 2014]:

$$\rho(C^i, C^j) = \frac{||(S^i)^{1/2}(U^i)^\top U^j (S^j)^{1/2}||^2_{\text{fro}}}{||\text{diag}S^i|| \cdot ||\text{diag}S^j||} \tag{14}$$

where $C^i = U^i S^i (U^i)^\top$ and $C^j = U^j S^j (U^j)^\top$ are their singular value decompositions. This measure ranges in $[0, 1]$. It is 0 if and only if C^i, C^j specify two orthogonal linear subspaces, and 1 if and only if C^i is a multiple of C^j.

In order to compare the difficulties of the permuted and disjoints MNIST experiments, we selected four datasets: $D_{original}, D_{permuted}, D_{5to9}$ and D_{0to4}, where $D_{original}$ is the original MNIST dataset; $D_{permuted}$ is the whole MNIST dataset but the pixels of every image is shuffled by the same randomly generated permutation; D_{5to9} consists of only the images of digits 5 to 9 from the MNIST dataset, and D_{0to4} has only the images of digits 0 to 4. Then for each of these four datasets, we computed a conceptor from the raw input images data inside it. The results were four conceptors $C_{original}, C_{permuted}, C_{5to9}$ and C_{0to4}.

In the permuted MNIST experiment, the network has to learn to recognize $D_{permuted}$ and $D_{original}$ sequentially, the overlap between their corresponding conceptors can be measured by $\rho(C_{original}, C_{permuted})$, which is around 0.3 on average.

In contrast, $\rho(C_{5to9}, C_{0to4})$ is much higher (≈ 0.95), which means the input images in D_{5to9} and D_{0to4} span roughly the same linear subspaces of the input memory space. Therefore, if a network is first trained on D_{5to9} and then on D_{0to4}, only a very small amount of components of the images in D_{0to4} can be exploited to compensate the errors, namely those components preserved by $F_{5to9} := \neg C_{5to9}$, whereas the linear transformation of the components inside the subspace characterized by C_{5to9} will be fixed. Figure 3 shows some images from D_{0to4} projected by C_{5to9} and F_{5to9}. Note that when learning to recognize the second dataset, CAB only allows the network to adjust its performance based on the projected versions displayed in the right column, which are much less legible than the images projected by C_{5to9}, shown in the left column.

Since the images in Figure 3 are also included in $D_{original}$, for comparison, we also visualized their components inside the linear subspaces characterized by

$C_{permuted}$ and $F_{permuted}$, which can be found in Figure 4. In the setting of permuted MNIST experiment, after the network is trained on $D_{permuted}$, it can only rely on the components displayed in the right column of Figure 4 to compensate its output errors. However, it is clear that the images in the right column of Figure 4 are more distinguishable than those in the right column of Figure 3, hence the permuted MNIST experiment is easier for CAB than the disjoint one.

A direction for future work, suggested by the analysis above, is to improve CAB such that the weights of the network can be adjusted even when the input patterns of different tasks lie in the same linear subspace. On the other hand, such similarity between tasks might be exploited to save the training time. So another question for further investigation is how to turn the similarity between different tasks into a desirable property rather than difficulty for continual learning.

Acknowledgments. The work reported in this article was partly funded through the European H2020 collaborative project NeuRAM3 (grant Nr 687299).

References

Robert M French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135, 1999.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *International Conference on Learning Representations*, 2014.

Herbert Jaeger. Controlling recurrent neural networks by conceptors. *Jacobs University Technical Reports*, (31), 2014. https://arxiv.org/abs/1403.3369.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017. http://www.pnas.org/content/114/13/3521.abstract.

Dharshan Kumaran, Demis Hassabis, and James L McClelland. What learning systems do intelligent agents need? complementary learning systems theory updated. *Trends in Cognitive Sciences*, 20(7):512–534, 2016.

Yann LeCun, Corinna Cortes, and Christopher JC Burges. The MNIST database of handwritten digits. *http: //yann. lecun. com/ exdb/ mnist/*, 1998.

Sang-Woo Lee, Jin-Hwa Kim, Jung-Woo Ha, and Byoung-Tak Zhang. Overcoming catastrophic forgetting by incremental moment matching. *Computing Research Repository*, 1703.08475, 2017. `http://arxiv.org/abs/1703.08475`.

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of Learning and Motivation*, 24:109–165, 1989.

Roger Ratcliff. Connectionist models of recognition memory: Constraints imposed by learning and forgetting functions. *Psychological Review*, 97(2):285–308, 1990.

Ari Seff. Implementation of “Overcoming catastrophic forgetting in neural networks” in tensorflow. *GitHub repository* ***https://github.com/ariseff/overcoming-catastrophic***, 2017.

Rupesh K Srivastava, Jonathan Masci, Sohrob Kazerounian, Faustino Gomez, and Juergen Schmidhuber. Compete to compute. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2310–2318. Curran Associates, Inc., 2013a. `http://papers.nips.cc/paper/5059-compete-to-compute.pdf`.

Rupesh K Srivastava, Jonathan Masci, Sohrob Kazerounian, Faustino Gomez, and Jürgen Schmidhuber. Compete to compute. *Advances in Neural Information Processing Systems*, pages 2310–2318, 2013b.