Continual Learning by Conceptor Regularization

Xu He
Department of Computer Science and Electrical Engineering
Jacobs University Bremen
Bremen 28759, Germany
x.he@jacobs-university.de

Abstract

Conceptor-aided backpropagation (CAB) can alleviate catastrophic forgetting in a feed-forward network. It uses conceptors to characterize the input space occupied by previous tasks, and projects the gradients to free space for learning new tasks, thus they do not interfere with the old tasks. In this paper, we discuss the relationship between conceptors and weight matrices trained using $L2$ regularization, which clarifies why and how CAB works. We also show that when the input spaces of different tasks overlap significantly, CAB will fail to learn new tasks due to vanished gradients. To overcome this drawback, we propose another regularization method based on conceptors. Empirical results confirm that the proposed approach can learn a classifier incrementally even when the input spaces of different tasks completely coincide.

1 Introduction

Conceptor-aided backpropagation (CAB) [He and Jaeger, 2018] was recently proposed to overcome catastrophic forgetting in feed-forward networks and has shown promising performance on several continual learning tasks based on the MNIST dataset [LeCun et al., 1998, Parisi et al., 2018]. CAB uses conceptors to characterize the linear subspaces of representations already used by previous tasks. During the backpropagation procedure for learning new tasks, CAB exploits only the components outside the already occupied space, thus learning new tasks does not interfere with the old tasks.

For many machine learning systems, however, different tasks may share the same input space but require distinct outputs. For example, an agent that needs to solve several vision-based tasks always receives natural images as input. Therefore, the input distributions for all these tasks are exactly the distribution of natural images. In this case, training such agent using CAB is difficult, since the input vectors of new tasks are completely inside the space occupied by previous tasks, and almost no input components are available for further adaptation.

In this paper, we modify the original approach of CAB and propose another regularization method based on conceptors. Instead of shielding the weight matrices by the free space in the pre-synaptic layer, our regularization method constrains them by the free space of the post-synaptic layer. As a result, new activation triggered by the change of parameters appears only in the free space, thus it does not interfere with old tasks. Moreover, useful features from the old tasks can be used by the new tasks, which allows a forward transfer learning in the network [Lopez-Paz et al., 2017].

To test our method empirically and demonstrate that it can overcome the drawback of CAB, we designed an incremental classifier learning experiment, in which a multi-class classification task is divided into a sequence of One-vs-All [Rifkin and Klautau, 2004] binary classification tasks. Experiment results show that, with the proposed method, a multilayer perceptron (MLP) is able to incrementally learn the binary classification subtasks and achieve a final multi-label accuracy that matches the performance of a jointly trained network.

2 Conceptor and L2 Regularization

In this section, we discuss the relationship between conceptors and L2 regularization, which will help understand how CAB and the proposed method work.

Conceptor Consider a training dataset of input-output pairs $D = f(x_1, y_1), \ldots (x_n, y_n)g$ where $x_i \; 2 \; \mathsf{R}^N, y_i \; 2 \; \mathsf{R}^M$. A conceptor matrix defined in [Jaeger, 2014] is a regularized identity map C that minimizes $L_C := \sum_{i=1}^{n} jjx_i \quad Cx_i jj^2 + \alpha \; ^2 jjCjj^2_{\text{fro}}$, and has a closed form solution as $C = XX^>(XX^> + \alpha \; ^2 I) \; ^1$, where X is a $N \quad n$ data collection matrix whose i-th column is x_i.

Suppose the singular value decomposition (SVD) of X has the form $X = UDV^>$ with U being an $N \quad N$ orthogonal matrix and V being an $n \quad N$ matrix with orthonormal columns. D is a diagonal matrix with descending diagonal entries $d_i \quad 0$. The SVD of C can be written in terms of the SVD of X: $C = U\Sigma U^T$, where Σ is a diagonal matrix with descending entries $\sigma_i = \frac{d_i^2}{d_i^2 + \; -2} \; 2 \; [0, 1]$. Therefore, C can be considered as a soft projection matrix. For vectors x in the linear subspace spanned by principal components (columns of U) with large variance d_i^2, $Cx \quad x$; for vectors x in the linear subspace spanned by principal components with low variance, $Cx \quad 0$.

Finally, conceptors are subject to most laws of Boolean logic, given the following operations:

$$: C := I \quad C, \tag{1}$$

$$C^i \_ C^j := (X^i X^{i>} + X^j X^{j>})(X^i X^{i>} + X^j X^{j>} + \alpha \; ^2 I) \; ^1 \tag{2}$$

$$C^i \wedge C^j := : (: C^i \_ : C^j) \tag{3}$$

Here, $: C$ characterizes the (soft) orthogonal complement of the subspace characterized by C. $C^i \_ C^j$ is computed from the union of two sets of input vectors collected as columns in X^i and X^j, it describes the direct sum of the linear spaces characterized by C^i and C^j. These two operations play an important role in CAB: given a conceptor C that describes the subspace already used by previous tasks, $: C$ returns the conceptor characterizing the free space; $C^i \_ C^j$ is useful for accumulating conceptors from sever tasks such that only one conceptor is kept to represent the space used by all previous tasks.

Relationship to Ridge Regression A linear map W that fits the dataset D by minimizing $\sum_{i=1}^{n} jjy_i \quad Wx_i jj^2$ is given by ridge regression: $W := YX^>(XX^>) \; ^1$, where Y is the $M \quad n$ output data collection matrix whose i-th column is y_i. If we add a $L2$ regularization term, and minimize $\sum_{i=1}^{n} jjy_i \quad W_{L2} x_i jj^2 + \alpha \; ^2 jjW_{L2} jj^2_{\text{fro}}$ instead, we obtain the ridge regression solution: $W_{L2} := YX^>(XX^> + \alpha \; ^2 I) \; ^1$. It is easy to see that $W_{L2} = W(XX^>)(XX^> + \alpha \; ^2 I) \; ^1 = WC$. Hence, ridge regression is a composition of conceptor projection and linear regression, and

$$8x \; 2 \; \mathsf{R}^N, Cx \quad 0 \; =) \quad W_{L2} x \quad 0 \tag{4}$$

This implication can also be observed when W_{L2} is a weight matrix inside a deep network trained with weight decay [Krogh and Hertz, 1992] (please refer to Appendix A for more details).

3 Methods

For simplicity, we illustrate the ideas of both CAB and the proposed method by only focusing on the scenario of sequentially training two supervised tasks on a 3-layer MLP. A complete algorithm of conceptor regularization that generalizes to any number of tasks and layers is given in Appendix B.

CAB Let $T^1 = f(x_i^1, y_i^1)g_{i2I^1}, T^2 = f(x_i^2, y_i^2)g_{i2I^2}$ be two tasks to be trained on a MLP with $L =$ 3 layers: $8l \; 2 \; f0, \ldots, L \quad 1g, a_l = [\hat{a}_l^>, 1]^>, \hat{a}_{l+1} = f_l(W_l a_l)$. Here W_l, f_l are weights and element-wise nonlinear transfer functions respectively, and $\hat{a}_0 := x$ is the input vector. CAB trains the network on the first task by minimizing the objective function $J^1 := L(\hat{a}_L(x^1, fW_l g_{l2f0;:::;L \quad 1g}), y^1) + \lambda \sum_{l=0}^{L \quad 1} jjW_l jj^2_{\text{fro}}$. Here λ is a regularizer parameter and L is some loss function depending on the network output $\hat{a}_L$ and the target y^1 of task 1. After the network is trained on T^1, the resulting weights that minimize the loss are saved as $fW_l^1 g_{l2f0;:::;L \quad 1g}$ and a batch of input vectors from $fx_i^1 g_{i2I^1}$ are fed into the trained network to collect a set of activation vectors $fa_{l\,i}^1 g$ from each layer

l. From these activation vectors, a conceptor C_l^1 is computed to characterize the linear subspace S_{C_l} already used by the first task in layer l. Importantly, C_l^1 preserves the activation vectors when the inputs are from T^1: $C_l^1 a_{l\,i}^1 \approx a_{l\,i}^1$, and its negation $F_l^1 := \neg C_l^1$ that characterizes the orthogonal complement of S_{C_l} (which is considered the free space S_{F_l}) will null these vectors: $F_l^1 a_{l\,i}^1 \approx 0$.

When the second task comes, the new weights W_l^2 (which will later replace W_l^1) are computed by adding incremental weights W_l^{inc} to the old weights W_l^1 : $W_l^2 := W_l^1 + W_l^{\text{inc}}$. In the training process of the second task, W_l^1 will be fixed and only W_l^{inc} will be adapted to minimize $J^2 := L(\hat{a}_L(x^2, \{W_l^2\}_{l=0:\ldots;L-1}), y^2) + \lambda \sum_{l=0}^{L-1} \|W_l^{\text{inc}}\|_{\text{fro}}^2$, but subject to an additional constraint that $\|W_l^{\text{inc}} C_l^1\|_{\text{fro}}^2 \approx 0$. CAB enforces the constraint by setting W_l^{inc} to $W_l^{\text{inc}} F_l^1$ at every step of the gradient descent, which essentially projects the rows of W_l^{inc} to the the orthogonal complement of S_{C_l}, thus $W_l^{\text{inc}} C_l^1 \approx 0$. As a result, the final weight W_l^2 does not forget the first task: $\forall l \in \{0, \ldots, L-1\}, f_l(W_l^2 a_l^1) = f_l((W_l^1 + W_l^{\text{inc}}) a_l^1) = f_l(W_l^1 a_l^1 + W_l^{\text{inc}} a_l^1) \approx f_l(W_l^1 a_l^1 + W_l^{\text{inc}} C_l^1 a_l^1) \approx f_l(W_l^1 a_l^1 + 0) = f_l(W_l^1 a_l^1)$.

Limitation Although this method works well on tasks with different input spaces, it will fail when their input spaces overlap significantly. Consider an extreme case where two tasks have exactly the same inputs but different outputs: $\{x_i^1\}_{i \in I^1} = \{x_i^2\}_{i \in I^2}$, $\{y_i^1\}_{i \in I^1} \neq \{y_i^2\}_{i \in I^2}$ (for example, $\{x_i^1\}_{i \in I^1}$ is the entire set of MNIST digit images, y_i^1, y_i^2 are binary classification outputs such that $y_i^1 = 1$ only if x_i^1 is an image of digit 0 and $y_i^2 = 1$ only if x_i^2 is an image of digit 1). In this case, it is possible to train both tasks on the same network with a multi-head output (i.e., using a separate matrix W_{L-1}^j at the last layer for each task j), which is commonly used in continual learning models (Bakker and Heskes [2003], Rusu et al. [2016], Li and Hoiem [2016], Serrà et al. [2018]). However, CAB will perform poorly in this case because enforcing $W_l^{\text{inc}} C_l^1 \approx 0$ will render W_l^{inc} useless. This can be shown by induction: since the input vectors of both tasks are the same, $a_0^2 = a_0^1 \in S_{C_0^1}$, $\forall l \in \{0, \ldots, L-2\}$, if $a_l^2 = a_l^1$, $a_{l+1}^2 = [f(W_l^1 a_l^2 + W_l^{\text{inc}} a_l^2)^\top, 1]^\top \approx [f(W_l^1 a_l^1 + W_l^{\text{inc}} C_l^1 a_l^1)^\top, 1]^\top \approx [f(W_l^1 a_l^1)^\top, 1]^\top = a_{l+1}^1$. Therefore, no layers except the last one (the multi-head output) will contribute to improving the network's performance on the second task.

Conceptor Regularization In order to overcome the above-mentioned problem, we replace the original constraint $\|W_l^{\text{inc}} C_l^1\|_{\text{fro}}^2 \approx 0$ by $\Omega(W_l^1, W_l^2, C_{l+1}^1) := \mathbb{E}[\|C_{l+1}^1 (f_l(W_l^2 \epsilon_l) - f_l(W_l^1 \epsilon_l))\|_2^2] \approx 0$. Here ϵ_l is a random vector that has the same shape as a_l and is drawn from the standard normal distribution. This new constraint is then expressed as a regularization term[1] to be added to the original objective function

$$J^2 = L(\hat{a}_L(x^1, \{W_l\}_{l=0:\ldots;L-1}), y^1) + \lambda \sum_{l=0}^{L-1} \|W_l^{\text{inc}}\|_{\text{fro}}^2 + \gamma \sum_{l=0}^{L-2} \Omega(W_l^1, W_l^2, C_{l+1}^1) \quad (5)$$

The intuition behind this regularization term is that changing W_l^1 to W_l^2 should not result in any change in the already used space $S_{C_{l+1}}$of the next layer. This forces the change caused by the increment weight W_l^{inc} to only happen in the free space $S_{F_{l+1}}$ of the next layer.

Formally, for any a_l, if $C_{l+1}^1 (f_l(W_l^2 a_l) - f_l(W_l^1 a_l)) \approx 0$, by Equ. 4, it implies that $W_{l+1}^1 (f_l(W_l^2 a_l) - f_l(W_l^1 a_l)) \approx 0$. One can then check by induction that with the new parameters $\{W_l^2\}_{l=0;1}$ and the task-specific output weight W_2^1, the 3-layer MLP does not forget the first task:

$$\begin{aligned}
f_2(W_2^1 f_1(W_1^2 f_0(W_0^2 x_i^1))) = \quad & f_2(W_2^1 [f_1(W_1^2 f_0(W_0^2 x_i^1)) - f_1(W_1^1 f_0(W_0^2 x_i^1)) + f_1(W_1^1 f_0(W_0^2 x_i^1))]) \\
\approx f_2(W_2^1 f_1(W_1^1 f_0(W_0^2 x_i^1))) = \quad & f_2(W_2^1 f_1(W_1^1 [f_0(W_0^2 x_i^1) - f_0(W_0^1 x_i^1) + f_0(W_0^1 x_i^1)])) \\
\approx f_2(W_2^1 f_1(W_1^1 f_0(W_0^1 x_i^1)))
\end{aligned}$$

On the other hand, as long as the old task does not occupy the entire space of the $(l+1)$-th hidden layer (in other words, C_{l+1}^1 is not almost the identity matrix), the regularization term does not imply $f_l(W_l^2 a_l) = f_l(W_l^1 a_l)$, thus W_l^{inc} can still be adapted. A way to check how much capacity is left for further learning is to plot the singular value spectra of C_{l+1}^1, if there are still some singular values

[1]This regularization is not applied to the last layer, since a saparate W_{L-2}^j is used for every task j.

Table 1: Accuracies on 10 Binary MNIST Task using different methods

Method	Testing Accuracy
Joint Training	98.18% (best achieved in five runs)
CAB ($\alpha = 6$)	23.74%
CAB ($\alpha = 10^{\ 7}$)	49.16%
CAB ($\alpha = 0.15$)	69.73% (best achieved in five runs)
Only Last Layer	64.59%
Proposed Method ($\alpha = 4$)	$98.032(\ \ 0.02)\%$ (averaged over five runs)

close to 0, $S_{C^1_{l+1}}$ does not equal to the entire space and the network is capable of learning more. In the case that the entire space of a hidden layer is occupied, the capacity can be extended by introducing new neurons in the overloaded hidden layer.

4 Experiments

Here we present only the results of the experiments. Please refer to Appendix C for more details.

10 One-vs-All MNIST To clearly demonstrate that the proposed method can overcome the limitation of CAB, we turned the standard task of classifying MNIST digits into a sequence of 10 binary classification subtasks, where every task is trained on the images from the entire MNIST training set, but for every new task, the network has to detect a different class by returning 1 for images from that class and 0 otherwise. After all 10 binary classification tasks are learned, we evaluate the network by its performance on classifying the entire test set into 10 classes, where the class with the largest output value is taken to be the prediction.

A standard classifier was trained as a control experiment, and we call this scheme "Joint Training". An ideal continual learning method should achieve an accuracy close to this case. For CAB, we chose different values for the aperture α. Larger aperture implies stronger protection of parameters for previous tasks. Since the last layer of the network is a multi-head operation, every task has its own output weights. For this reason, we also conducted an experiment where only the last layer can be adapted after the first task is learned. Table 1 compares the proposed method with other methods.

Disjoint MNIST In the second experiment, we split the original MNIST dataset into two disjoint subsets each containing images of five digits ($0 \quad 4$ in the first dataset and $5 \quad 9$ in the second) and compute the average accuracy after learning them one after the other. In this case, the input spaces of two tasks overlap with each other but they are not exactly the same. The best result on this experiment was 99.0% achieved by Serrà et al. [2018]. With the same network architecture, conceptor regularization achieves comparable performance (99.05%).

Permuted MNIST Finally, we tested the conceptor regularization method on the permuted MNIST experiment, in which we randomly shuffle the pixels in MNIST to create another dataset of the same size as the original MNIST, and compute the average performance after learning them one after the other. In this case, due to the random permutation, input spaces of two different tasks intersect very little, so CAB was able to achieve very high accuracy: 97.34%. Conceptor regularization performs almost as good as CAB with a final accuracy at 97.3%.

5 Conclusion

Conceptor-aided backprop fails to learn new tasks when the input spaces of different tasks overlap significantly. We proposed another regularization method based on conceptors to overcome this limitation. To illustrate how and why our method works, we also discussed the relationship between conceptors and L2 regularization. Conventionally, $L2$ regularization has been used to prevent a model from learning noise in training data in order to achieve better generalization. CAB and conceptor regularization show that the capacity saved by regularization from learning one task can be squeezed out by conceptors for learning another task.

Acknowledgement The work reported in this article was funded through the European Horizon 2020 project NeuRAM3 (grant 687299). The author would like to thank Herbert Jaeger and one of the anonymous reviewers for their valuable feedbacks.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

Bart Bakker and Tom Heskes. Task clustering and gating for bayesian multitask learning. *Journal of Machine Learning Research*, 4(May):83–99, 2003.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

Xu He and Herbert Jaeger. Overcoming catastrophic interference using conceptor-aided backpropagation. In *International Conference on Learning Representations*, 2018.

Herbert Jaeger. Controlling recurrent neural networks by conceptors. Technical Report 31, Jacobs University Technical, 2014. `https://arxiv.org/abs/1403.3369`.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

Anders Krogh and John A Hertz. A simple weight decay can improve generalization. In *Advances in Neural Information Processing Systems*, pages 950–957, 1992.

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

Zhizhong Li and Derek Hoiem. Learning without forgetting. In *European Conference on Computer Vision*, pages 614–629. Springer, 2016.

David Lopez-Paz et al. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, pages 6470–6479, 2017.

German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *arXiv preprint arXiv:1802.07569*, 2018.

Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5(Jan):101–141, 2004.

Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

Joan Serrà, Dídac Surís, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. *arXiv preprint arXiv:1801.01423*, 2018.

A Conceptor and Weight Decay

When W_{L2} is a weight matrix in a deep network trained with weight decay, the implication $\forall x \in \mathbb{R}^N, Cx = 0 \Rightarrow W_{L2}x = 0$ cannot be shown analytically, but it is possible to check this empirically: the statement holds if W_{L2} nulls vectors in the linear subspace spanned by principal components (PCs) corresponding to very low variances. In other words, if we compute the matrix $\tilde{W}_{L2}$ that represents the same linear transformation as W_{L2} but changes the basis of input space to the PCs: $\tilde{W}_{L2} := W_{L2}U$, the norms of the last columns in $\tilde{W}_{L2}$ should be close to 0.

To verify our hypothesis, we trained two feed-forward networks with $[784 \quad 800 \quad 800 \quad 10]$ neurons on the MNIST classification task, one with weight decay and the other one without. We call the weight matrices connecting the two layers of hidden neurons W_{L2} and W, respectively. After training the network, we computed the principal components of activations in the first layer of hidden neurons and collected these PCs as columns of U and U_{L2}, respectively. Figure 1 visualizes the norms of columns in $\tilde{W} = WU$ and $\tilde{W}_{L2} = W_{L2}U$. It is easy to see that with $L2$ regularization, the columns corresponding to PCs with low variances have almost zero norms, hence this agrees with the nulling effect of conceptors.

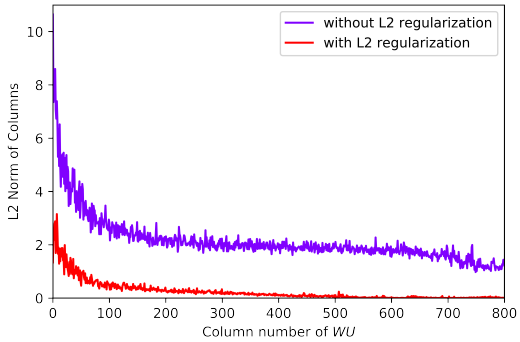

Figure 1: The L2 norm of columns in $\tilde{W} := WU$ and $\tilde{W}_{L2} := W_{L2}U$. Due to weight decay, the norm of the last columns of $\tilde{W}_{L2}$ are close to 0, hence vectors in the linear subspace spanned by principal components corresponding to small variances will be nulled by it.

B Conceptor Regularization Algorithm

The following algorithm describes how to train a network with L layers on a sequence of J supervised tasks using conceptor regularization ($\forall j = 1, \ldots, J, T^j = f(x_i^j, y_i^j)_{i=1,\ldots}$):

Initialization (no task trained yet):
$\forall l = 1, \ldots, L-1$, set A_l^0 to zero. $\forall l = 0, \ldots, L-1$, set W_l^0 to zero and randomly initialize W_l^{inc}.

Incremental task learning: For $j = 1, \ldots, J$ do:

1. Let $W_l^j := W_l^{j-1} + W_l^{\text{inc}}$ for $l < L$ and $W_L^j := W_L^{\text{inc}}$. Update W_l^{inc} by stochastic gradient descent to minimize $J = L(a_L(x^j, f W_l^j)_{l=1,\ldots,L-1}), y^j) + \lambda \sum_{l=0}^{L-1} \|W_l^{\text{inc}}\|_{\text{fro}}^2 + \gamma \sum_{l=0}^{L-2} \Omega(W_l^{j-1}, W_l^j, A_l^{j-1})$. The gradient of J with respect to W_l^{inc} is estimated using a batch of samples for x^j, y^j and ϵ_l for $l = 0, \ldots, L-2$. The samples for ϵ_l are randomly drawn from standard normal distribution.
2. After training on the j-th task, run the forward procedure again on a batch of input vectors from the j-th training dataset, and collect activations $\{a_{l,i}^j\}$ to compute a conceptor C_l^j.
3. Update the conceptor for already used space using OR operation: $\forall l = 1, \ldots, L-1, A_l^j = A_l^{j-1} \_ C_l^j$