Sequential Covariance-Matrix Estimation with Application to Mitigating Catastrophic Forgetting

Tomer Lancewicki*, Ben Goodrich†, Itamar Arel‡
Department of Electrical Engineering and Computer Science
University of Tennessee
Email: *tlancewi@utk.edu, †bgoodric@utk.edu, ‡itamar@ieee.org

***Abstract*—Catastrophic forgetting is a problem encountered with neural networks as well as other learning systems whereby past representations are lost as new representations are learned. It has been shown that catastrophic forgetting can be mitigated in neural networks by using a neuron selection technique, dubbed "cluster-select," which performs online clustering over the network inputs to partition the network such that only a subset of neurons are used for a given input vector. Cluster-select can benefit by using Mahalanobis distance which relies on an inverse covariance estimate. Unfortunately, covariance estimation is problematic when lacking a very large number of samples relative to the number of input dimensions. One way to tackle this problem is through the use of a shrinkage estimator that offers a compromise between the sample covariance matrix and a well-conditioned matrix with the aim of minimizing the mean-squared error (MSE). In online environments, such as those in which catastrophic forgetting can occur, data arrives sequentially, requiring the covariance matrix to be estimated sequentially. Therefore, in this work we derive sequential update rules for the shrinkage estimator and approximate it's related inverse. The online covariance estimator is applied to the cluster-select technique with results that demonstrate further improvements in terms of effectively mitigating catastrophic forgetting.**

I. Introduction

Catastrophic forgetting is a well studied problem with neural networks and other learning systems where past representations are lost as new representations are learned. This problem is particularly severe with models that have a shared global pool of resources where updates that are meant to affect one region of the input space can affect regions that are distant.

Catastrophic forgetting can have detrimental effects to real world applications of deep learning. One such area is in training very large networks on very large datasets. It was shown in [1] that there is a law of diminishing returns for increasing the capacity of a network by adding more neurons. This suggests that as training datasets grow larger, it may become difficult to avoid under-fitting when learning all of the details of the dataset. Catastrophic forgetting may be involved here due to the network being unable to retain all of the characteristics of a large dataset.

Another area where catastrophic forgetting can adversely affect performance is online learning where the input examples are provided in a non-stationary manner. In this type of problem, the input examples can be highly correlated which precipitates catastrophic forgetting. A recent example appears in [2], where the authors successfully applied reinforcement learning along with a deep neural network to successfully learn to play many different Atari games. A key element to their success was the use of a technique known as "Experience Replay". The use of a replay buffer decorrelates the input examples by storing past observations in a large buffer and presenting them as training examples in a random order.

Clearly, better techniques to mitigate catastrophic forgetting need to be developed in order for neural networks to be scaled up to more complex real world problems. In this paper we extend a neuron selection technique dubbed "cluster-select" which was previously introduced in [3]. The aim of cluster-select is to reduce activation overlap by selectively choosing neurons which are used in the feed forward and back propagation pass.

The general framework of cluster-select is to assign each neuron a cluster in addition to its regular weights. When an input is fed in; l out of k neurons which have the nearest centroids are selected. A sub-network is built out of the l neurons such that only those neurons are used in the feed-forward pass. Selecting neurons in this manner has the effect of partitioning the input space such that different regions are assigned to different neurons, making the representation explicit such that it no longer overlaps. In essence, the network is divided in the sense that not all neurons are active at the same time.

In this work, we extend this technique by sequentially estimating a covariance for each centroid which allows the use of Mahalanobis distance instead of Euclidean distance. The use of Mahalanobis distance allows the centroids to take covariance into consideration when covering the input space. A performance boost is demonstrated in a non-stationary catastrophic forgetting test case.

In environments where data arrives sequentially, the covariance matrix is required to be updated sequentially [4]. Techniques such as cross-validation, to achieve regularization, or model selection are not feasible [5]. Instead, to avoid complexity, it is often assumes that the covariance matrix is known in advance [6] or that it is restricted to a specific model, such as a diagonal matrix [7], [8]. Moreover, when the number of observations n is comparable to the number of variables p the covariance estimation problem becomes more challenging. In such scenarios, the sample covariance matrix is not well-conditioned nor is it necessarily invertible (despite the fact that those two properties are required for most applications). When $n \leq p$, the inversion cannot be computed at all [9, Sec. 2.2].

An extensive body of literature concerning improved estimators in such situations exist [10], [11]. In the absence of a specific knowledge about the structure of the true covariance matrix, the most successful approach so far has, arguably, been shrinkage estimation [12]. It has been demonstrated in [13] that the largest sample eigenvalues are systematically biased upwards, and the smallest ones downwards. This bias is corrected by pulling down the largest eigenvalues and pushing up the smallest ones, towards the grand mean of all sample eigenvalues. This is an application of the general shrinkage principle, going back to [14], [15].

This paper contains a fundamental work, providing a sequential update of the shrinkage estimator in the sense of mean-squared error (MSE), which can be obtained analytically. In addition to the catastrophic forgetting motivating case, this is a general result that can be utilized in a wide range of machine learning applications. The paper elaborates on the Ledoit Wolf shrinkage estimator that addresses the problem of covariance matrix estimation when the number of samples is relatively small compared to the number of variables. The estimator offers a compromise between the sample covariance matrix and a well-conditioned matrix (also known as the *target*) with the aim of minimizing the MSE. We derived a sequential update rule for the shrinkage estimator. The estimator is propagated using recursive update equations each time new data is available. The paper is organized as follows: Section 2 presents the general idea of the shrinkage estimator. In Section 3 we derived a sequential update for the shrinkage estimator. Finally, in Section 4 we conduct an experiment which combines the sequential covariance estimation technique with the cluster-select network architecture.

II. Shrinkage Estimator for Covariance Matrices

Notations: we denote vectors in lowercase boldface letters and matrices in uppercase boldface. The transpose operator is denoted as $(\cdot)^T$. The trace, the determinant and the Frobenius norm of a matrix are denoted as $\mathrm{Tr}(\cdot)$, $|\cdot|$ and $\|\cdot\|_F$, respectively. The identity matrix is denoted as $\mathbf{I}$, while $\mathbf{e} = [1, 1, \ldots, 1]^T$ is a column vector of all ones, and $\mathbf{1} = \mathbf{e}\mathbf{e}^T$ is a matrix of ones. The centering matrix denoted as $\mathbf{H} = \mathbf{I} - \mathbf{1}/n$. For any real matrices $\mathbf{R}_1$ and $\mathbf{R}_2$, the inner product is defined as $\langle \mathbf{R}_1, \mathbf{R}_2 \rangle = \mathrm{Tr}\left(\mathbf{R}_1^T \mathbf{R}_2\right)$, where $\langle \mathbf{R}_1, \mathbf{R}_1 \rangle = \|\mathbf{R}_1\|_F^2$ [16, Sec. 2.20]. To make for easier reading, when $\mathbf{R}_1$ is a random matrix, we use the notation $V(\mathbf{R}_1) = E\left\{\|\mathbf{R}_1 - E\{\mathbf{R}_1\}\|_F^2\right\}$ (the sum of variances of the elements in $\mathbf{R}_1$).

We briefly review single-target shrinkage estimator by following [13], [17], which is generally applied to high-dimensional estimation problems. Let $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$ be a data matrix of size $p \times n$ where $\{\mathbf{x}_i\}_{i=1}^n$ is a sample of *independent identical distributed* (i.i.d.) p-dimensional vectors drawn from a density with a mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. The sample mean $\mathbf{m}_n$ is defined as

$$\mathbf{m}_n = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i. \tag{1}$$

When the number of observations n is large (i.e., $n \gg p$), the most common estimator of $\boldsymbol{\Sigma}$ is the sample covariance matrix

$$\mathbf{S}_n = \frac{1}{n-1} \sum_{i=1}^{n} (\mathbf{x}_i - \mathbf{m}_n)(\mathbf{x}_i - \mathbf{m}_n)^T. \tag{2}$$

Both $\mathbf{S}_n$ and $\mathbf{m}_n$ are unbiased estimators of $\boldsymbol{\Sigma}$ and $\boldsymbol{\mu}$, respectively, i.e., $E\{\mathbf{S}_n\} = \boldsymbol{\Sigma}$ and $E\{\mathbf{m}_n\} = \boldsymbol{\mu}$. The shrinkage estimator $\hat{\boldsymbol{\Sigma}}(\lambda_n)$ is in the form

$$\hat{\boldsymbol{\Sigma}}(\lambda_n) \quad = \quad (1 - \lambda_n)\mathbf{S}_n + \lambda_n \mathbf{T}_n \tag{3}$$

where the target $\mathbf{T}_n$ is a restricted estimator of $\boldsymbol{\Sigma}$ defined as

$$\mathbf{T}_n = \frac{\mathrm{Tr}(\mathbf{S}_n)}{p} \mathbf{I}. \tag{4}$$

The objective is to find an estimator $\hat{\boldsymbol{\Sigma}}(\lambda_n)$ which minimizes the *mean squared error* (MSE)

$$E\left\{ \left\| \hat{\boldsymbol{\Sigma}}(\lambda_n) - \boldsymbol{\Sigma} \right\|_F^2 \right\}. \tag{5}$$

The value of λ_n that minimize the MSE (5) is defined as

$$\lambda_{On} = \arg\min_{\lambda_n} E\left\{ \left\| \hat{\boldsymbol{\Sigma}}(\lambda_n) - \boldsymbol{\Sigma} \right\|_F^2 \right\} \tag{6}$$

and can be given by the distribution-free formula

$$\lambda_{On} = \frac{E\{\langle \mathbf{T}_n - \mathbf{S}_n, \boldsymbol{\Sigma} - \mathbf{S}_n \rangle\}}{E\left\{ \|\mathbf{T}_n - \mathbf{S}_n\|_F^2 \right\}}. \tag{7}$$

The scalar λ_{On} is called the oracle shrinkage coefficient, since its depends on the unknown covariance matrix $\boldsymbol{\Sigma}$. Therefore, λ_{On} (7) must be estimated.

A. Estimations of the Oracle Shrinkage Coefficient λ_{On} (7)

It has been shown in [18, Sec. 3.B] that the target $\mathbf{T}_n$ (4) is a private case of the general target framework which allows to reformulate λ_{On} (7) as

$$\lambda_{On} = \frac{V(\mathbf{S}_n) - V(\mathbf{T}_n)}{E\left\{ \|\mathbf{T}_n - \mathbf{S}_n\|_F^2 \right\}}. \tag{8}$$

The oracle shrinkage coefficient λ_{On} (8) can be estimated from its sample counterparts as

$$\hat{\lambda}_{On} = \max\left(\min\left(\frac{\hat{V}(\mathbf{S}_n) - \hat{V}(\mathbf{T}_n)}{\|\mathbf{T}_n - \mathbf{S}_n\|_F^2}, 1 \right), 0 \right), \tag{9}$$

where the symbol ^ indicates an estimated value of the parameter. The estimated oracle shrinkage coefficient $\hat{\lambda}_{On}$ (9) is bounded in [0,1] in order to keep the shrinkage estimator $\hat{\boldsymbol{\Sigma}}\left(\hat{\lambda}_{On}\right)$ positive-definite as required from a covariance matrix [18, Sec. 2.A]. As in [18], we use the unbiased estimators $\hat{V}(\mathbf{S}_n)$ and $\hat{V}(\mathbf{T}_n)$ (i.e., $E\left\{\hat{V}(\mathbf{S}_n)\right\} = V(\mathbf{S}_n)$ and $E\left\{\hat{V}(\mathbf{T}_n)\right\} = V(\mathbf{T}_n)$) that are equal to

$$\hat{V}(\mathbf{S}_n) = \frac{n}{(n-1)^2 (n-2)} \left(\nu(n) - \frac{(n-1)^2}{n} \|\mathbf{S}_n\|_F^2 \right) \tag{10}$$

and

$$\hat{V}(\mathbf{T}_n) = \frac{n}{(n-1)^2 (n-2)\, p} \left(\nu(n) - \frac{(n-1)^2}{n} \mathrm{Tr}^2(\mathbf{S}_n) \right), \tag{11}$$

respectively, where $\nu(n)$ is equal to

$$\nu(n) = \sum_{i=1}^{n} \left(\|\mathbf{x}_i - \mathbf{m}_n\|_F^2 \right)^2. \tag{12}$$

In the rest of the paper we derived the sequential update rule $\hat{\mathbf{\Sigma}}(\lambda_n)$ (3), and an update rule that approximate the oracle shrinkage coefficient $\hat{\lambda}_{On}$ (9) when a new observation $\mathbf{x}_{n+1}$ is added, using only the current knowledge of $\mathbf{S}_n$, $\mathbf{m}_n$ and n. Similar derivations can be done when an observation $\mathbf{x}_n$ is removed.

III. Sequential Update of the Shrinkage Estimator

We want to know what happens to $\hat{\mathbf{\Sigma}}\left(\hat{\lambda}_{On}\right)$ (3) when we add an observation $\mathbf{x}_{n+1}$, using only the current knowledge of $\mathbf{S}_n$, $\mathbf{m}_n$ and n. Setting $\mathbf{d}_{n+1} = \mathbf{x}_{n+1} - \mathbf{m}_n$ while using [16, 15.12.(c)], we have the following update rules for $\mathbf{m}_n$ (1) and $\mathbf{S}_n$ (2) when an observation $\mathbf{x}_{n+1}$ is added

$$\mathbf{m}_{n+1} = \mathbf{m}_n + \frac{1}{n+1}\mathbf{d}_{n+1} \tag{13}$$

$$\mathbf{S}_{n+1} = \frac{n-1}{n}\mathbf{S}_n + \frac{1}{n+1}\mathbf{d}_{n+1}\mathbf{d}_{n+1}^T. \tag{14}$$

Based on $\mathbf{S}_{n+1}$ (14), we can write the update rule for the target $\mathbf{T}_n$ (4) as

$$\mathbf{T}_{n+1} = \frac{n-1}{n}\mathbf{T}_n + \frac{1}{(n+1)\, p} \|\mathbf{d}_{n+1}\|_F^2 \mathbf{I} \tag{15}$$

By using $\mathbf{S}_{n+1}$ (14) and $\mathbf{T}_{n+1}$ (15), the update rule for the shrinkage estimator $\hat{\mathbf{\Sigma}}(\lambda_n)$ (3) can be written as

$$\hat{\mathbf{\Sigma}}(\lambda_{n+1}) = \mathbf{G}_n + \mathbf{F}_n \tag{16}$$

where $\mathbf{G}_n$ and $\mathbf{F}_n$ defined as

$$\mathbf{G}_n = \frac{n-1}{n}\hat{\mathbf{\Sigma}}(\lambda_n) + (1-\lambda_{n+1})\frac{1}{n+1}\mathbf{d}_{n+1}\mathbf{d}_{n+1}^T \tag{17}$$

and

$$\mathbf{F}_n = \frac{1}{(n+1)\, p}\lambda_{n+1} \|\mathbf{d}_{n+1}\|_F^2 \mathbf{I} + \frac{n-1}{n}(\lambda_n - \lambda_{n+1})(\mathbf{S}_n - \mathbf{T}_n), \tag{18}$$

respectively.

The expression $\hat{\mathbf{\Sigma}}(\lambda_{n+1})$ (16) is general for any arbitrary values of λ_{n+1} and λ_n. In the following section we derived the update rule for the estimated oracle shrinkage coefficient $\hat{\lambda}_{On}$ (9).

A. Update Rules for the Oracle Shrinkage Coefficient Estimator $\hat{\lambda}_{On}$ (9)

The expression $\hat{\lambda}_{On}$ (9) involves update rules for its numerator and denominator. The update rule for $\hat{\lambda}_{On}$ (9) denominator is

$$\|\mathbf{T}_{n+1} - \mathbf{S}_{n+1}\|_F^2 = \left(\frac{n-1}{n}\right)^2 \|\mathbf{T}_n - \mathbf{S}_n\|_F^2 + \zeta_n \tag{19}$$

where

$$\zeta_n = \frac{1}{(n+1)^2}\left(1 - \frac{1}{p}\right)\|\mathbf{d}_{n+1}\|_F^4 + \frac{2(n-1)}{n(n+1)}\mathbf{d}_{n+1}^T(\mathbf{S}_n - \mathbf{T}_n)\mathbf{d}_{n+1}. \tag{20}$$

The numerator of $\hat{\lambda}_{On}$ (9) involves an update of $\hat{V}(\mathbf{S}_n)$ (10) and $\hat{V}(\mathbf{T}_n)$ (11), which in turn involve the update rules for $\|\mathbf{S}_n\|_F^2$, $\mathrm{Tr}^2(\mathbf{S}_n)$ and $\nu(n)$ (12).

In the same manner as (19), the update rules of $\|\mathbf{S}_n\|_F^2$ and $\mathrm{Tr}^2(\mathbf{S}_n)$ can be written as

$$\|\mathbf{S}_{n+1}\|_F^2 = \left(\frac{n-1}{n}\right)^2 \|\mathbf{S}_n\|_F^2 + \frac{1}{(n+1)^2}\|\mathbf{d}_{n+1}\|_F^4 + \frac{2(n-1)}{n(n+1)}\mathbf{d}_{n+1}^T\mathbf{S}_n\mathbf{d}_{n+1} \tag{21}$$

and

$$\mathrm{Tr}^2(\mathbf{S}_{n+1}) = \left(\frac{n-1}{n}\right)^2 \mathrm{Tr}^2(\mathbf{S}_n) + \frac{1}{(n+1)^2}\|\mathbf{d}_{n+1}\|_F^4 + \frac{2(n-1)}{n(n+1)}\mathrm{Tr}(\mathbf{S}_n)\|\mathbf{d}_{n+1}\|_F^2 \tag{22}$$

respectively. The update rule for $\nu(n)$ (12) is more complicated. As we will show next, the current knowledge of $\mathbf{S}_n$, $\mathbf{m}_n$ and n allow us to give only approximated update rule for $\nu(n)$ (12).

B. Approximated $\nu(n)$ Update Rule

Until now we derived precise update rules when a new observation $\mathbf{x}_{n+1}$ is arrived, while using only the current knowledge of $\mathbf{S}_n$, $\mathbf{m}_n$ and n. However, this is not the case for $\nu(n)$ update rule . The full derivation of $\nu(n)$ (12) update rule appears in Appendix A and is equal to

$$\nu(n+1) = \nu(n) + \phi_n - \frac{4}{n+1}\mathbf{d}_{n+1}^T\mathbf{X}\mathbf{H}\boldsymbol{\kappa} \tag{23}$$

where

$$\phi_n = \frac{n(n^3+1)}{(n+1)^4}\|\mathbf{d}_{n+1}\|_F^4 + \frac{4(n-1)}{(n+1)^2}\mathbf{d}_{n+1}^T\mathbf{S}_n\mathbf{d}_{n+1} + \frac{2(n-1)}{(n+1)^2}\|\mathbf{d}_{n+1}\|_F^2\mathrm{Tr}(\mathbf{S}_n) \tag{24}$$

and

$$\boldsymbol{\kappa} = diag\left(\mathbf{H}\mathbf{X}^T\mathbf{X}\mathbf{H}\right) \tag{25}$$

The last term on the right hand side of $\nu(n+1)$ (23) can not be computed as a function of only $\mathbf{S}_n$ and $\mathbf{d}_{n+1}^T$. However, since

$$\mathbf{d}_{n+1}^T \mathbf{X}\mathbf{H}\boldsymbol{\kappa} = \sum_{i=1}^{n} (\mathbf{x}_{n+1} - \mathbf{m}_n)^T (\mathbf{x}_i - \mathbf{m}_n) \|\mathbf{x}_i - \mathbf{m}_n\|_F^2, \tag{26}$$

we can use the distribution of the sample mean $\mathbf{m}_n$ (1) and the central limit theorem [19, Ch. 8.3] to show that

$$\begin{aligned} &\lim_{n\to\infty} \sum_{i=1}^{n} (\mathbf{x}_{n+1} - \mathbf{m}_n)^T (\mathbf{x}_i - \mathbf{m}_n) \|\mathbf{x}_i - \mathbf{m}_n\|_F^2 \\ &= \sum_{i=1}^{n} (\mathbf{x}_{n+1} - \boldsymbol{\mu})^T (\mathbf{x}_i - \boldsymbol{\mu}) \|\mathbf{x}_i - \boldsymbol{\mu}\|_F^2 \end{aligned} \tag{27}$$

Since the new observation $\mathbf{x}_{n+1}$ is independent of previous observations $\mathbf{x}_i$, $i = 1, \ldots, n$, it holds that

$$E\left\{\lim_{n\to\infty} \mathbf{d}_{n+1}^T \mathbf{X}\mathbf{H}\boldsymbol{\kappa}\right\} = 0. \tag{28}$$

Therefore, a reasonable approximation for $\nu(n+1)$ (23) will be

$$\tilde{\nu}(n+1) = \tilde{\nu}(n) + \phi_n. \tag{29}$$

For completeness, we provide in Appendix B an upperbound for $\nu(n+1)$ (23) based on the current values of $\nu(n)$ (12), $\mathbf{S}_n$ (2) and $\mathbf{d}_{n+1}$. The use of the approximated update rule $\tilde{\nu}(n)$ (29) instead of $\nu(n)$ (23) will result with an approximation of $\hat{\lambda}_{On}$ (9) denoted as

$$\tilde{\lambda}_{On} = \max\left(\min\left(\frac{\tilde{V}(\mathbf{S}_n) - \tilde{V}(\mathbf{T}_n)}{\|\mathbf{T}_n - \mathbf{S}_n\|_F^2}, 1\right), 0\right), \tag{30}$$

where $\tilde{V}(\mathbf{S}_n)$ and $\tilde{V}(\mathbf{T}_n)$ are calculated in the same manner as $\hat{V}(\mathbf{S}_n)$ (10) and $\hat{V}(\mathbf{T}_n)$ (11), respectively, where $\nu(n)$ (23) is replaced by its approximation $\tilde{\nu}(n)$ (29).

IV. Experiments

In this section, we run experiments to evaluate the effectiveness of incorporating covariance estimation in cluster-select. We begin in Section IV-A by providing a brief overview of the cluster-select framework and discuss the general design of the experiments. Note that a more in depth discussion of this technique can be found in [20]. In Section IV-B we perform experiments on the MNIST [21] handwritten digit dataset, and in Section IV-C we perform experiments on a dataset from a gas sensor array measuring gas mixtures [22].

A. Test Setup

The main premise behind cluster-select is to reduce activation overlap which causes catastrophic forgetting. To accomplish this reduction in overlap, only a subset of the neurons within a hidden layer are active at any given time. To determine which neurons are active, each neuron i is given a centroid vector $\mathbf{c_i}$ in addition to its weight vector. For a given input vector denoted by $\mathbf{x}_{n+1}$, the distance from that input to neuron i's centroid can be computed using Mahalanobis distance.

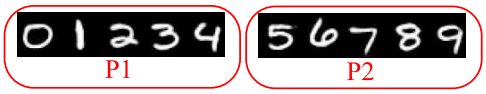

Figure 1. How MNIST was split into $P1$ and $P2$

$$\delta_i = \sqrt{(\mathbf{x}_{n+1} - \mathbf{c_i})^T \boldsymbol{\Sigma}_i^{-1} (\mathbf{x}_{n+1} - \mathbf{c_i})} \tag{31}$$

where $\boldsymbol{\Sigma}_i$ is the unknown covariance matrix and therefore must be estimated. Previously, Euclidean distance was used for this measure, i.e., $\boldsymbol{\Sigma}_i$ replaced by the identity matrix $\mathbf{I}$. However, with the covariance estimation technique, a measure of a centroid i's covariance can be estimated sequentially, allowing for Mahalanobis distance to be used in this test. Only the l nearest neurons are allowed to have a nonzero output. All other neuron outputs are forced to zero, which essentially creates a sub-network consisting of l neurons, and partitions the network such that different regions of the input space activate different neurons.

Our technique is compared to maxout [23] and local winner-take-all (LWTA) [24] networks, which have been shown to help with catastrophic forgetting in [25]. These types of networks also partition the input space such that only a subset of neurons are active at any given time. A software tool known as hyperopt [26] was used to find the optimal hyper-parameters (number of neurons, learning rate, number of centroids selected, etc) for the test cases discussed below. In terms of computation, this technique scales well to parallel architectures since computing the distances for all neurons can be performed simultaneously.

B. MNIST Experiment

For the first catastrophic forgetting test, we used the MNIST handwritten digit dataset. An autoencoder [27] was constructed consisting of a hidden layer with 50 activations which was used to reduce MNIST to 50 dimensions. The dataset was then divided into two subsets as illustrated in Figure 1. Samples that had class labels for digits 0 through 4 were placed in subset $P1$, and samples for which the class was 5 through 9 were placed in subset $P2$. A network was trained on $P1$ to predict which of the 5 classes the sample belonged. Once training on $P1$ was complete (no improvement on test error was observed for 30 epochs), the dataset was switched to $P2$. After switched to $P2$, both $P1$ and $P2$ error were observed for forgetting.

Figure 2 below depicts a log-scale possibilities frontiers plot of $P1$ test error vs. $P2$ test error during the forgetting phase. This type of plot was first introduced in [25] and provides a way to visually compare the $P1$ and $P2$ errors. A curve that is close to the bottom left hand corner indicates that the network was able to learn both $P1$ and $P2$ at some point during training. This figure illustrates a definite improvement of using Mahalanobis distance over Euclidean because it shows that the network was able to capture more of $P1$ without misclassifying $P2$. Forgetting curves for maxout and LWTA networks

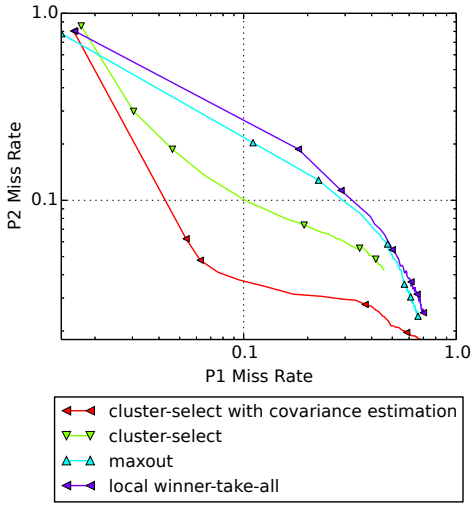

Figure 2. Cluster-Select MNIST Result

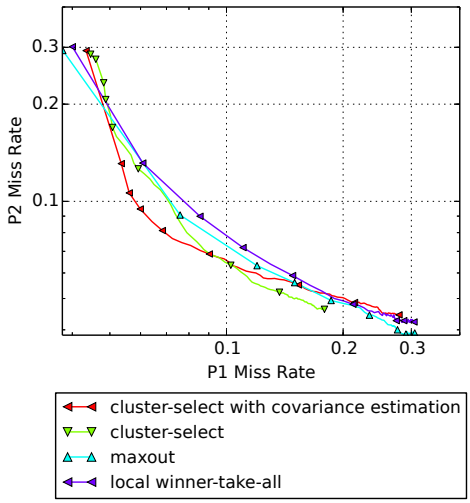

Figure 3. Cluster-Select Gas Sensor Array Dataset Result

are also shown. The results depicted in Figure 2 demonstrate that cluster-select using euclidean distance outperforms both LWTA and maxout networks. Adding covariance to cluster-select provides a significant boost in performance.

C. Experiment with Gas Sensor Array Dataset

For the second catastrophic forgetting test, a dataset was utilized that consists of readings from a gas sensor array under dynamic gas mixtures. This dataset has 19 dimensions, including a time dimension which was removed. The task is to determine which of two gas concentrations (Ethylene and CO, or Methane and Ethylene) are present, hence creating a binary classification task. The dataset was first normalized such that each dimension had zero mean and unit variance. Then, to make the dataset more challenging, Gaussian noise was added with variance 1.0.

To test catastrophic forgetting, dataset $P1$ was scrambled such that the input dimensions were randomly rearranged, producing dataset $P2$. Just as before, training is performed on $P1$ until it reaches satisfactory performance (no improvement observed on the test set for 30 epochs). Training is then switched to $P2$ and both $P1$ and $P2$ are observed.

The results for the gas dataset follow the same trends as before, cluster-select outperforms both LWTA and maxout. Adding covariance further boosts performance. Cluster-select with covariance estimation is generally closer to the bottom left hand corner of the graph indicating better performance.

V. Conclusions

A key challenge in our approach to mitigating catastrophic forgetting stems from the need to obtain the covariance matrix of the data, which is unknown in practice and should be estimated. In order to avoid additional complexity during the training process, it is commonly assumed that the covariance matrix is known in advance, or alternatively, simplified estimators are employed. In this work, we derived a sequential update rule for the shrinkage estimator that offers a compromise between the sample covariance matrix and a well-conditioned matrix with the aim of minimizing the mean-squared error (MSE). The optimal solution is obtained analytically, which provides a notable advantage since it reduces the complexity involved in estimating covariance. The analytical solution is an alternative to the computationally complex *cross-validation* (CV) procedure for establishing the values of shrinkage coefficients. Experimental results demonstrate that incorporating a covariance estimate into the neuron clustering scheme provides a notable boost in performance. The proposed estimator is fits well to our clustering approach, since it outperforms the sample covariance matrix, while introducing fewer hyper-parameters than alternative covariance estimation schemes.

Appendices

A. $\nu\,(n)(12)$ Update Rule Derivation

We can write $\nu\,(n+1)$ as

$$\nu\,(n+1) = \sum_{i=1}^{n+1} \left(\left\| \mathbf{x}_i - \mathbf{m}_{n+1} \right\|_F^2 \right)^2$$

$$= \left(\left\| \mathbf{x}_{n+1} - \mathbf{m}_{n+1} \right\|_F^2 \right)^2 + \sum_{i=1}^{n} \left(\left\| \mathbf{x}_i - \mathbf{m}_{n+1} \right\|_F^2 \right)^2$$

$$= \left(\frac{n}{n+1} \right)^4 \left\| \mathbf{d}_{n+1} \right\|_F^4 + \left\| diag\left(\mathbf{A}^T \mathbf{A} \right) \right\|_F^2 \qquad (32)$$

where

$$\mathbf{A} = \mathbf{X}\mathbf{H} - \frac{1}{n+1} \mathbf{d}_{n+1} \mathbf{e}^T. \qquad (33)$$

Then, by using $\boldsymbol{\kappa}$ (25), we can write the last term in (32) as

$$\left\| \boldsymbol{\kappa} - \frac{2}{n+1} \mathbf{H}\mathbf{X}^T \mathbf{d}_{n+1} + \frac{1}{(n+1)^2} \mathbf{e} \left\| \mathbf{d}_{n+1} \right\|_F^2 \right\|_F^2$$

$$= \boldsymbol{\kappa}^T \boldsymbol{\kappa} - \frac{4}{n+1} \boldsymbol{\kappa}^T \mathbf{H}\mathbf{X}^T \mathbf{d}_{n+1} + \frac{2}{(n+1)^2} \left\| \mathbf{d}_{n+1} \right\|_F^2 \boldsymbol{\kappa}^T \mathbf{e}$$

$$+\frac{4}{(n+1)^2}\mathbf{d}_{n+1}^T\mathbf{X}\mathbf{H}\mathbf{X}^T\mathbf{d}_{n+1}$$

$$-\frac{4}{(n+1)^3}\left\|\mathbf{d}_{n+1}\right\|_F^2\mathbf{d}_{n+1}^T\mathbf{X}\mathbf{H}\mathbf{e}+\frac{n}{(n+1)^4}\left\|\mathbf{d}_{n+1}\right\|_F^4$$

$$=\upsilon(n)+\frac{4(n-1)}{(n+1)^2}\mathbf{d}_{n+1}^T\mathbf{S}_n\mathbf{d}_{n+1}+\frac{n}{(n+1)^4}\left\|\mathbf{d}_{n+1}\right\|_F^4$$

$$+\frac{2}{(n+1)^2}\left\|\mathbf{d}_{n+1}\right\|_F^2\mathrm{Tr}\left(\mathbf{S}_n\right)\mathbf{e}-\frac{4}{n+1}\boldsymbol{\kappa}^T\mathbf{H}\mathbf{X}^T\mathbf{d}_{n+1}$$

$$-\frac{4}{(n+1)^3}\left\|\mathbf{d}_{n+1}\right\|_F^2\mathbf{d}_{n+1}^T\mathbf{X}\mathbf{H}\mathbf{e} \tag{34}$$

where the last term in (34) is equal zero (since $\mathbf{X}\mathbf{H}\mathbf{e}=\mathbf{0}$). Then, by substituting (34) into $\nu(n+1)$ (32) we get the final update rule of $\nu(n+1)$ (23).

B. An Upper Bound for $\nu(n+1)$ (23)

An upper bound of $\nu(n+1)$ can be calculated as follows: By using [16, 12.5.(b)] we can write

$$\begin{aligned}\left|\mathbf{d}_{n+1}^T\mathbf{X}\mathbf{H}\boldsymbol{\kappa}\right| &= \sqrt{\mathrm{Tr}\left(\mathbf{d}_{n+1}^T\mathbf{X}\mathbf{H}\boldsymbol{\kappa}\mathbf{d}_{n+1}^T\mathbf{X}\mathbf{H}\boldsymbol{\kappa}\right)}\\ &= \sqrt{\mathrm{Tr}\left(\left(\boldsymbol{\kappa}\mathbf{d}_{n+1}^T\mathbf{X}\mathbf{H}\right)\left(\boldsymbol{\kappa}\mathbf{d}_{n+1}^T\mathbf{X}\mathbf{H}\right)\right)}\\ &\leq \sqrt{\mathrm{Tr}\left(\left(\boldsymbol{\kappa}\mathbf{d}_{n+1}^T\mathbf{X}\mathbf{H}\right)\left(\boldsymbol{\kappa}\mathbf{d}_{n+1}^T\mathbf{X}\mathbf{H}\right)^T\right)}\\ &= \sqrt{\mathrm{Tr}\left(\boldsymbol{\kappa}\mathbf{d}_{n+1}^T\mathbf{X}\mathbf{H}\mathbf{X}^T\mathbf{d}_{n+1}\boldsymbol{\kappa}^T\right)}\\ &= \sqrt{(n-1)\,\mathbf{d}_{n+1}^T\mathbf{S}_n\mathbf{d}_{n+1}\nu(n)}.\end{aligned}$$

Therefore, an upper bound $\hat{\nu}(n+1)$ for $\nu(n+1)$, i.e., $\hat{\nu}(n+1)\geq\nu(n+1)$, is equal to

$$\hat{\nu}(n+1)=\tilde{\nu}(n+1)+\frac{4}{n+1}\sqrt{(n-1)\,\mathbf{d}_{n+1}^T\mathbf{S}_n\mathbf{d}_{n+1}\nu(n)} \tag{35}$$

where $\tilde{\nu}(n+1)$ defined in (29).

ACKNOWLEDGMENT

This work has been supported in part by the DARPA/MTO Cortical Processor program. The opinions in this paper are those of the authors and do not necessarily reflect the opinions of the funding agency.

REFERENCES

[1] Y. Dauphin and Y. Bengio, "Big neural networks waste capacity," *CoRR*, vol. abs/1301.3583, 2013.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[3] B. Goodrich and I. Arel, "Neuron clustering for mitigating catastrophic forgetting in feedforward neural networks," in *2014 IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments (CIDUE)*, pp. 62–68.

[4] G. S. Babu and S. Suresh, "Meta-cognitive neural network for classification problems in a sequential learning framework," *Neurocomputing*, vol. 81, pp. 86 – 96, 2012.

[5] J. F. de Freitas, M. Niranjan, and A. H. Gee, "Regularisation in sequential learning algorithms," in *Advances in Neural Information Processing Systems*, 1998, pp. 458–464.

[6] L. Xu, J. Wang, and Q. Chen, "Kalman filtering state of charge estimation for battery management system based on a stochastic fuzzy neural network battery model," *Energy Conversion and Management*, vol. 53, no. 1, pp. 33 – 39, 2012.

[7] E. Manitsas, R. Singh, B. Pal, and G. Strbac, "Distribution system state estimation using an artificial neural network approach for pseudo measurement modeling," *IEEE Transactions on Power Systems*, vol. 27, no. 4, pp. 1888–1896, Nov 2012.

[8] S. Young, J. Lu, J. Holleman, and I. Arel, "On the impact of approximate computation in an analog destin architecture," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 5, pp. 934–946, May 2014.

[9] E. Theodorou, J. Buchli, and S. Schaal, "A generalized path integral control approach to reinforcement learning," *J. Mach. Learn. Res.*, vol. 11, pp. 3137–3181, Dec. 2010.

[10] P. J. Bickel and E. Levina, "Regularized estimation of large covariance matrices," *The Annals of Statistics*, vol. 36, no. 1, pp. pp. 199–227, 2008.

[11] A. Rohde and A. B. Tsybakov, "Estimation of high-dimensional low-rank matrices," *Ann. Statist.*, vol. 39, no. 2, pp. 887–930, 04 2011.

[12] O. Ledoit and M. Wolf, "Nonlinear shrinkage estimation of large-dimensional covariance matrices," *Institute for Empirical Research in Economics University of Zurich Working Paper*, no. 515, 2011.

[13] ——, "A well-conditioned estimator for large-dimensional covariance matrices," *Journal of Multivariate Analysis*, vol. 88, no. 2, pp. 365 – 411, 2004.

[14] C. M. Stein, "In admissibility of the usual estimator for the mean of a multivariate normal distribution," *Proc. Third Berkeley Symp. Math. Statist. Probab*, vol. 1, pp. 197–206, 1956.

[15] W. James and C. Stein, "Estimation with quadratic loss," in *Proceedings of the fourth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 1961, 1961, pp. 361–379.

[16] G. A. F. Seber, *A Matrix Handbook for Statisticians*. John Wiley and Sons, Inc., 2007.

[17] J. Schafer and K. Strimmer, "A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics," *Statist. Appl. Genet. Molec. Biol.*, vol. 4, no. 1, 2005.

[18] T. Lancewicki and M. Aladjem, "Multi-target shrinkage estimation for covariance matrices," *IEEE Transactions on Signal Processing*, vol. 62, no. 24, pp. 6380–6390, Dec 2014.

[19] R. A. Johnson, *Statistics: principles and methods*. John Wiley & Sons, 1992.

[20] B. Goodrich and I. Arel, "Unsupervised neuron selection for mitigating catastrophic forgetting in neural networks," in *2014 IEEE 57th International Midwest Symposium on Circuits and Systems (MWSCAS)*, pp. 997–1000.

[21] Y. Lecun and C. Cortes, "The MNIST database of handwritten digits." [Online]. Available: http://yann.lecun.com/exdb/mnist/

[22] J. Fonollosa, S. Sheik, R. Huerta, and S. Marco, "Reservoir computing compensates slow response of chemosensor arrays exposed to fast varying gas concentrations in continuous monitoring," *Sensors and Actuators B: Chemical*, vol. 215, pp. 618–629, 2015.

[23] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," *arXiv preprint arXiv:1302.4389*, 2013.

[24] R. K. Srivastava, J. Masci, S. Kazerounian, F. Gomez, and J. Schmidhuber, "Compete to compute," in *Advances in Neural Information Processing Systems*, 2013, pp. 2310–2318.

[25] I. J. Goodfellow, M. Mirza, X. Da, A. Courville, and Y. Bengio, "An empirical investigation of catastrophic forgeting in gradient-based neural networks," *arXiv:1312.6211*, 2013.

[26] J. Bergstra, D. Yamins, and D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," in *Proceedings of The 30th International Conference on Machine Learning*, 2013, pp. 115–123.

[27] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.