Continual Learning with Distributed Optimization: Does CoCoA Forget under Task Repetition?

Martin Hellkvist, Ayça Özcelikkale, Anders Ahlén
Department of Electrical Engineering, Uppsala University, Sweden
martin.hkvist@gmail.com, {Ayca.Ozcelikkale, Anders.Ahlen}@angstrom.uu.se

Abstract—We focus on the continual learning problem where the tasks arrive sequentially and the aim is to perform well on the newly arrived task without performance degradation on the previously seen tasks. In contrast to the continual learning literature focusing on the centralized setting, we investigate the distributed estimation framework. We consider the well-established distributed learning algorithm CoCoA. Our results illustrate the importance of task repetition for continual learning with CoCoA. In particular, even when there is a shared solution for all tasks, task repetition may be necessary for satisfactory performance. We show how the dimensions of the offline and centralized problem affects the learning performance substantially; even though CoCoA processes only a subset of features and data samples at each iteration.

I. INTRODUCTION

When presented with a stream of data, continual learning [1], [2] is the act of learning from new data while not forgetting what was learnt previously. New data can, for instance, come from a related classification task with new fine-grained classes, or it can have statistical distribution shift compared to the previously seen data. Each set of data that is presented to the model is referred to as a task. Continual learning aims to create models which perform well on all seen tasks without the need to retrain from scratch when new data comes [1]–[4].

The central issue in continual learning is forgetting, which measures the performance degradation on previously learned tasks as new tasks are learnt by the model [2]–[4], see (4) in Section II-A for a formal definition. If a model performs poorly on old tasks, it is said to exhibit catastrophic forgetting [2]–[4]. To achieve continual learning, the forgetting must be reduced. Nevertheless, the key enabling factors for achieving continual learning are not well established, even for linear models [2], [3].

Forgetting is closely related to the error performance under non-stationary distributions. Various phenomena of interest, such as financial time-series and target tracking, often exhibit structural changes in signal characteristics over time. Hence, performance under non-stationary distributions has been studied in a number of scenarios; including with random drift in the unknown within a distributed learning setting [5], [6] as well under switching system dynamics [7]–[9].

We consider the continual learning problem from a distributed learning perspective, where optimization is performed over a network of computational nodes. In addition to supporting scalability, distributed learning is also attractive for scenarios where the data is already distributed over a network, for example in sensor networks [10]–[13], or in dictionary learning where sub-dictionaries are naturally separated over the network [14], or in multi-task settings where the nodes have separate but related tasks [15], [16].

In this work, we focus on the well-established distributed learning algorithm CoCoA [17]–[22], which allows the nodes to use a local solver of their choice for their local subproblems. In contrast to the setting of [15], [16] where the individual nodes have different tasks but these tasks do not change over time, here all nodes have the same task and this task changes over time for all nodes. We investigate the ability of CoCoA to perform continual learning for a linear model. Task repetition, which has been ignored to a large extent in [22], is an important aspect of our setting. We illustrate the performance of the algorithm with varying number of tasks, data points and number of repetitions of the tasks. revealing the trade-offs between the forgetting, the convergence performance and the dimensions of both the offline centralized problem, i.e., the problem of solving all tasks in one large batch in a centralized manner, and the nodes’ local problems. Our main findings can be summarized as follows:

- CoCoA does not forget if all the tasks are under-parametrized and there is a shared solution that solves all the tasks. (Lemma 2)
- Large improvements in the learning performance of the algorithm can be obtained through task repetition under a shared solution (Fig. 1-2 vs. Fig. 3-4) but this is not guaranteed when there is no shared solution (Fig. 7).
- The learning performance of the algorithm is directly affected by the dimensions of the offline centralized problem despite the fact that the algorithm processes only a subset of features and data samples at each iteration: If the total number of samples over all tasks is close to the total number of unknown parameters, then the algorithm may have slow convergence and relatively large error. (Section III-B)

II. PROBLEM STATEMENT

A. Definition of a Task

The data for task $\tau(t)$ consists of the feature matrix $A_{\tau(t)} \in \mathbb{R}^{n_{\tau(t)} \times p}$, and the corresponding vector of outputs $y_{\tau(t)} \in \mathbb{R}^{n_{\tau(t)} \times 1}$. We focus on fitting a linear model such that,

$$y_{\tau(t)} = A_{\tau(t)}w_{\tau(t)}$$

(1)
where \( w_{\tau(t)} \in \mathbb{R}^{p \times 1} \) is the vector of unknown model parameters. We refer to the above regression problem with the data \((A_{\tau(t)}, y_{\tau(t)})\) as task \( \tau(t) \).

### B. Continual Learning

We would like to fit the linear model in (1) to a sequence of tasks,

\[
S = \{(A_{\tau(t)}, y_{\tau(t)})\}_{t=1}^T.
\]

At a given time instant \( t \), we have access to \((A_{\tau(t)}, y_{\tau(t)})\) but not to the other tasks. Hence, we would like to solve all tasks in \( S \) when we have access to only one of them at a given time. Throughout the paper, we apply the following assumption on the tasks \((A_{\tau(t)}, y_{\tau(t)})\).

**Assumption 1.** There exists a solution \( w_* \in \mathbb{R}^{p \times 1} \) that solves all the tasks in \( S \), i.e.,

\[
\exists w_* \in \mathbb{R}^{p \times 1} : A_{\tau(t)} w_* = y_{\tau(t)}, \quad t = 1, \ldots, T.
\]

Note that Assumption 1 can be interpreted as a stationarity assumption. This constitutes a reasonable scenario in the case of highly overparameterized models, i.e., models that have more tunable parameters than data samples, and has been used in the centralized continual learning setting of [3]. This setting is also closely related to no-regret online learning in distributed settings [23], [24], where typically the data points are distributed over the network instead of the individual features of the unknown parameter as in our setting with CoCoA. We consider the investigation of the setting of Assumption 1 as good starting point for the general case since it facilitates a tractable analysis. However, it is not always natural to make this assumption, hence we study a setting where Assumption 1 does not hold in Section V.

For a given parameter vector \( w \), we measure the forgetting by the squared loss on a specific task \((A_{\tau(t)}, y_{\tau(t)})\), i.e.,

\[
\frac{1}{n_{\tau(t)}} \| A_{\tau(t)} w - y_{\tau(t)} \|^2,
\]

where \( \| \cdot \| \) denotes the \( \ell_2 \)-norm, as done in previous works [3], [25]. While training over the sequence of tasks \( S \), a corresponding sequence of parameter vectors estimates is generated, denoted by \( w_t, t = 1, \ldots, T \). We denote the forgetting of \( w_t \) as the forgetting of the tasks up until and including the current task,

\[
\mathcal{F}_S(t) = \frac{1}{t} \sum_{i=1}^{t} \frac{1}{n_{\tau(i)}} \| A_{\tau(i)} w_t - y_{\tau(i)} \|^2.
\]

Note that forgetting is defined in terms of data fit for \( y_{\tau(t)} \) instead of estimation accuracy for \( w \).

### C. Distributed Continual Learning with CoCoA

In CoCoA, the unknowns are distributed over a network of \( K \) nodes [17]–[20]. The unknown vector \( w \) is partitioned according to the partitioning \( P = \{p_k\}_{k=1}^K \), such that node \( k \) governs \( p_k \) of the \( p \) unknowns in \( w \in \mathbb{R}^{p \times 1} \). Note that this is in contrast to settings where the observations are distributed over the network. The partitioning consists of mutually exclusive sets of indices, hence each entry of the unknown vector (hence each column of a given feature matrix) is associated with only one node, i.e., \( \sum_{k=1}^K p_k = p \).

**Algorithm 1: Implementation of CoCoA** [17] for (5).

```
1 Input: Tasks \( S = \{(A_{\tau(t)}, y_{\tau(t)})\}_{t=1}^T \). Partitioning scheme \( P = \{p_k\}_{k=1}^K \), number of iterations \( T \) for CoCoA to run per task.
2 Initialize: \( w_0 = 0 \).
3 for \( t = 1, \ldots, T \) do
4 \( x^0_t = w_{t-1} \)
5 \( v_k^0 = K A_{\tau(t),[k]} w_{t-1} \) \( \forall k \).
6 for \( i = 0, 1, \ldots, T_c - 1 \) do
7 \( \tilde{v}_i = \frac{1}{K} \sum_{k=1}^K v_k^i \)
8 for \( k \in \{1, 2, \ldots, K\} \) do
9 \( \Delta x_k^i = \frac{1}{K} A_{\tau(t),[k]} (y_{\tau(t)} - \tilde{v}_i) \)
10 \( x_k^{i+1} = x_k^i + \Delta x_k^i \)
11 \( v_k^{i+1} = \tilde{v}_i + K A_{\tau(t),[k]} \Delta x_k^i \)
12 enddo
endfor
endfor
```

For each task \((A_{\tau(t)}, y_{\tau(t)})\), the columns of \( A_{\tau(t)} \in \mathbb{R}^{n_{\tau(t)} \times p} \) are partitioned according to the partitioning for the unknowns. We denote the matrix of columns associated with node \( k \) as \( A_{\tau(t),[k]} \). For each presented task \( \tau(t) \) in the sequence \( S \), we run CoCoA to minimize the forgetting of the current task, i.e.,

\[
\min_{w_t} \frac{1}{2} \| A_{\tau(t)} w_t - y_{\tau(t)} \|^2\quad \text{CoCoA is an iterative algorithm, which is run for } T \text{ iterations for each task. For notational clarity, the unknown vector is represented by } w_t \text{ for the outer iteration } t \text{ over the tasks and by } x_i \text{ for the inner CoCoA iteration } i.
\]

The partitioning of \( w_t \) and \( x_i \) which correspond to the local matrices \( A_{\tau(t),[k]} \) are denoted by \( w_{t,[k]} \) and \( x_{i,[k]} \), respectively. Hence the overall training scheme, presented in Algorithm 1, consists of three nested for-loops: the outer iterations \( t = 1, \ldots, T \) over the sequence of tasks; the inner iterations \( i = 0, \ldots, T_c - 1 \) of CoCoA; and the parallelization over the nodes \( k = 1, \ldots, K \).

For task \( \tau(t) \), the nodes iteratively update their local variables \( x_{i,[k]} \) by finding the update \( \Delta x_{i,[k]} \) as the solution to the following local subproblem in iteration \( i = 0, \ldots, T_c - 1 \), [17, Sec. 3.1]

\[
\min_{\Delta x_{i,[k]}} \frac{1}{2K n_{\tau(t)}} \| \tilde{v}_i - y_{\tau(t)} \|^2 + \frac{1}{2\sigma^t_{\tau(t)}} \| A_{\tau(t),[k]} \Delta x_{i,[k]} \|^2
\]

where \( \tilde{v}_i \) is the aggregated shared estimation of \( y_{\tau(t)} \) in iteration \( i \) and \( \sigma^t_{\tau(t)} \) is a hyperparameter for the algorithm. This is a convex problem in \( \Delta x_{i,[k]} \). Setting the gradient w.r.t. \( \Delta x_{i,[k]} \) to zero and choosing the minimum \( \ell_2 \)-norm solution, we obtain the solution as \( \Delta x_{i,[k]} = \frac{1}{\sigma^t_{\tau(t)}} A_{\tau(t),[k]}^T (y_{\tau(t)} - \tilde{v}_i) \), where \( (\cdot)^T \) denotes the Moore-Penrose pseudoinverse. Using the notation of [17], we set \( \sigma^t = \gamma K \), with \( \gamma \in (0, 1] \), as these are considered safe choices [20]. With these choices, these parameters cancel out to give the explicit expressions in Algorithm 1.

In order to utilize the knowledge from the previous tasks, we do the following: i) we initialize CoCoA with the parameter vector from the previous step, i.e., we set \( x^0 = w_{t-1} \); ii)
we set the initial point for the local contributions as $v^0_k = K A_{\tau(t),[k]} x^0_k$.

### D. Offline and centralized problem

Throughout our analysis of the distributed continual learning problem, we consider the offline and centralized problem as the reference problem, which corresponds to solving the unique set of tasks in $S$ simultaneously in a centralized fashion, i.e.,

$$y_S = A_S w, \quad y_S = \begin{bmatrix} y_1 \\ \vdots \\ y_M \end{bmatrix} \in \mathbb{R}^{N \times 1}, \quad A_S = \begin{bmatrix} A_1 \\ \vdots \\ A_M \end{bmatrix} \in \mathbb{R}^{N \times p},$$

where $\{(A_m, y_m)\}_{m=1}^M$ is the unique set of tasks in $S$, with $A_m \in \mathbb{R}^{n_m \times p}$ and $y_m \in \mathbb{R}^{n_m \times 1}$, and the total number of samples over all the unique tasks is $N = \sum_{m=1}^M n_m$. Throughout the paper, we use subindices $\tau(t)$ and $m$ to emphasize different task settings. The notation $\{(A_m, y_m)\}_{m=1}^M$ refers to the unique set of $M$ tasks in $S$, and is used when each task appears once. The notation $\tau(t) \in \mathbb{N}_+$, $t = 1, \ldots, T$, is used when tasks can be repeated.

### III. OVERPARAMETERIZED LOCAL PROBLEMS

If $p_k \geq n_{\tau(t)}$ for all $k = 1, \ldots, K$ and $t = 1, \ldots, T$, then the local problems are overparameterized, i.e., the nodes’ local feature matrices $A_{\tau(t),[k]} \in \mathbb{R}^{n_{\tau(t)} \times p_k}$ are broad. In this section, we assume that the matrices $A_{\tau(t),[k]}$ have full rank, i.e., $n_{\tau(t)}$, thus

$$A_{\tau(t),[k]}^+ = I_{n_{\tau(t)}}.$$

Using (7), the updates of $v_k^{i+1}$ for $i \geq 0$ (line 11 in Algorithm 1) can be rewritten as

$$v_k^{i+1} = \hat{v}^i + A_{\tau(t),[k]}^+ (y_{\tau(t)} - \hat{v}^i) = y_{\tau(t)}.$$

It follows that $\hat{v}^i = \frac{1}{K} \sum_{k=1}^K v_k^i = y_{\tau(t)}$, $i \geq 1$ and as a result, the steps $\Delta x_k^i$ for $i \geq 1$ are zero, i.e.,

$$\Delta x_k^i = \frac{1}{K} A_{\tau(t),[k]}^+ (y_{\tau(t)} - \hat{v}^i) = 0.$$

In other words, CoCoA in this setting converges after its first iteration $i = 0$. To fix the solution $x^{1}$ that CoCoA converges to, we insert the initial values of $x^{0}$ = $w_{t-1}$, and $v^0_k = KA_{\tau(t),[k]} w_{t-1}$, finding that

$$\hat{v}^0 = A_{\tau(t)} w_{t-1},$$

and that the local estimates $x_k^{1}$ in CoCoA converge to

$$x_k^{1} = \hat{v}^1 = \hat{v}^0 + \frac{1}{K} A_{\tau(t),[k]}^+ (y_{\tau(t)} - A_{\tau(t)} w_{t-1}).$$

Combining over the nodes $k = 1, \ldots, K$, the parameter vector $w_t = x^1$ after training on task $\tau(t)$ is,

$$w_t = P_{\tau(t)} w_{t-1} + A_{\tau(t)} y_{\tau(t)}.$$

where we have introduced the notation

$$P_{\tau(t)} = (I_p - A_{\tau(t)} A_{\tau(t)}^T);$$

$$A_{\tau(t)} = \frac{1}{K} \left[ (A_{\tau(t),[1]}^T \right]^T, \ldots, (A_{\tau(t),[K]}^T \right]^T \in \mathbb{R}^{p \times n_{\tau(t)}}.$$

In this setting, CoCoA always solves the latest seen task, i.e.,

$$\hat{y}_{\tau(t)} = A_{\tau(t)} w_{t}.$$

![Fig. 1: The size of the final step from $w_{T-1}$ to $w_T$ after training on each task once, versus the number of tasks $M$ and the number of samples per task $n_m$.](image1)

![Fig. 2: The forgetting of $w_T$ after training on each task once, versus the number of tasks $M$ and the number of samples per task $n_m$.](image2)
Fig. 3: The size of the final step from $w_{T-1}$ to $w_T$ after training on each task 1000 times, versus the number of tasks $M$ and the number of samples per task $n_m$.

Figure 1 and the forgetting in the final step $F_S(T)$ in Figure 2. We observe the following:

**Remark 1.** Even though CoCoA, i.e., the inner iterations of Algorithm 1, converge in one step and solves the latest seen task (see Lemma 1), the obtained solution $w_T$ after solving all tasks once can exhibit forgetting if the number of tasks $M$ or the number of data points per task $n_m$ is too small. The forgetting can be improved by increasing the number of tasks $M$ (hence increasing the number of data points) due to existence of the solution $w_*$ that solves all tasks.

**B. Cyclic sequence of tasks**

We now repeat the experiment of Section III-A, but extend the training sequence so that the model trains on each task 1000 times. In other words, with $M$ tasks we set the sequence as $\tau(t) = 1, \ldots, M, 1, \ldots, M, \ldots, t = 1, \ldots, T$, with $T = 1000M$. We plot the convergence in Figure 3 and the forgetting in Figure 4. The plots illustrate that the algorithm converges for most pairs of $(n_m, M)$, and that the forgetting for the final solution is low. However, the plots also illustrate that both the convergence and forgetting are worse along a streak of pairs $(n_m, M)$, for which $N \approx p$, i.e., where the total number of samples $N = \sum_{m=1}^{M} n_m$ is close to or equal to the number of parameters $p$, for example $(1, 160)$, $(2, 80)$ and $(4, 40)$. We now analyze this phenomenon, distinguishing between the following scenarios: i) $N > p$; ii) $N \approx p$; and iii) $N < p$. In the below, recall that $n_m$ is constant over $m = 1, \ldots, M$, hence $N = Mn_m$. Recall that we generate the matrices $A_m$ as independent standard Gaussian matrices, and as a result, the offline and centralized matrix $A_S$ has full rank with probability one.

i) $N > p$: If $M$ or $n_m$ is large enough so that the total number of samples $N = Mn_m$ exceeds the number of parameters $p$, then the offline and centralized problem $y_S = A_S w$ in (6) is underparameterized, i.e., there are less parameters than samples. The matrix $A_S \in \mathbb{R}^{N \times p}$ has full rank, and $y_S = A_S w_*$. Hence, $w_*$ is the unique solution to the offline and centralized problem. This suggests that $w_T = w_*$, since the solutions for large $M$ or $n_m$ in Figure 3 and 4 converge to a solution $w_T$ which solves all tasks, and $w_*$ is the only solution which does so.

ii) $N \approx p$: If the total number of samples $N = Mn_m$ is close to or equal to the number of parameters $p$, then the offline and centralized feature matrix $A_S$ is nearly square or square. The results in Figure 3 and 4 illustrate that Algorithm 1 does not converge, and does not solve the previously seen tasks. The non-convergence of the algorithm for these pairs of $(n_m, M)$ are further illustrated in Figure 5, where we plot the forgetting as a function of the outer iterations $t$ in Algorithm 1, i.e., $F_S(t)$, where we have evaluated the forgetting every 1000 iterations, and fixed the number of samples per task to $n_m = 2$. Figure 5 illustrates that if $M = 80$, then the vectors $w_t$ do not converge, and are instead diverging. If $M \approx 80$ but $M \neq 80$, then the convergence is very slow compared to the case with $M$ further away.

iii) $N < p$: In this scenario, the offline and centralized problem is overparameterized. Hence, there is possibly an infinite number of solutions $w$ which solve all the tasks simultaneously. In this regime, $w_t$ generally do not converge to $w_*$. In Figure 6, we plot the distance from $w_T$ to the solution $w_*$ versus the number of tasks $M$ for the experiment in Figure 3 and 4. Figure 6 illustrates that for small $M$, even though the final solution $w_T$ solves all the tasks, it is not equal to $w_* = 1$. Consistent with our observations for scenario i), Figure 6 suggests that $w_t$ converges to $w_*$ for large $M$.

**IV. UNDERPARAMETERIZED TASK**

A task $(A_{\tau(t)}, y_{\tau(t)})$ where $A_{\tau(t)} \in \mathbb{R}^{n_{\tau(t)} \times p}$ is such that $p < n_{\tau(t)}$ is referred to as underparameterized, as there is a smaller number of tuneable parameters in $w \in \mathbb{R}^{p \times 1}$ than the number of equations in $A_{\tau(t)} w = y_{\tau(t)}$. If the feature matrix $A_{\tau(t)}$ of an underparameterized task is full rank, then $w_*$ from Assumption 1 is the unique solution to that task. By [17, Theorem 2], a sufficiently large number of iterations $T_c$ of CoCoA guarantees an arbitrarily small optimality gap. In other words, if $T_c$ is large enough while learning an
underparameterized task \((A_{r(t)}, y_{r(t)})\), then the estimate \(x^T_o\) of CoCoA will converge to \(w_o\). Hence, we have the following lemma:

**Lemma 2.** If all tasks in the sequence \(S\) are underparameterized and have full rank feature matrices, and \(T_e \rightarrow \infty\), then Algorithm 1 converges to \(w_o\), which is the unique solver of all tasks in \(S\).

V. FAMILY OF TASKS WITHOUT ASSUMPTION 1

We now consider a setting where Assumption 1 does not hold. We again consider the cyclic manner of training on \(M\) tasks as in Section III-B. Tasks \((A_m, y_m)\) with even and odd indices \(m\) are generated with \(y_m = A_m w_{even}\) and \(y_m = A_m w_{odd}\), respectively. We set \(w_{even} = 1_p \times 1,\) and \(w_{odd} = [1, 0, 0.9, 1, 0.1, 0.1]T\). We randomly create the features matrices \(A_m\) as independent standard Gaussian matrices. We set \(n_m = 2\) samples and vary the number of tasks \(M\).

In Figure 7, we present the forgetting \(F_S(t)\) versus the number of outer iterations \(t\) of Algorithm 1, for different values of \(M\). These plots illustrate the different convergence behaviour for small and large \(M\). In particular, for small values of \(M\) (\(M = 2\)), the algorithm quickly converges to a solution which solves all the tasks. (Although \(w_{even} \neq w_{odd}\), there can be such a shared solution since the centralized, offline system is overparameterized for \(M\) small.) For larger values of \(M\) (\(M = 10\)), convergence is slower. For higher values of \(M\) (\(M = 150\)), the algorithm does not converge. Although not visible in Figure 7, the forgetting for \(M = 150\) fluctuates on the range \([4, 7]\) in the plot for \(t > 2000\).

These results illustrate that although the total number of samples for the two types of task with \(w_{even}\) and \(w_{odd}\) increases with \(M\), the forgetting does not necessarily improve. This is consistent with the fact that for larger values of \(M\) there is a higher number of equations that needs to be satisfied by the solution vector of CoCoA.

VI. CONCLUSIONS

We have quantified the learning performance improvement with task repetition for continual learning with the distributed learning algorithm CoCoA. Our results revealed that even though CoCoA processes only subsets of rows and columns of the system of equations formed by the offline and centralized problem, dynamics of the convergence and forgetting of the algorithm is significantly affected by the dimensions of this centralized. There are several interesting lines of future work, e.g., characterization of convergence and forgetting behaviour under other data models as well as development of methods to improve the learning behaviour of the algorithm.

REFERENCES

[1] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, “Continual lifelong learning with neural networks: A review,” *Neural Networks*, vol. 113, pp. 54–71, 2019.

[2] J. K. et al., “Overcoming catastrophic forgetting in neural networks,” *Proc. of the National Academy of Sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[3] I. Evron, E. Moroshko, R. Ward, N. Srebro, and D. Soudry, “How catastrophic can catastrophic forgetting be in linear regression?” in *Proc. of Thirty Fifth Conf. on Learning Theory*, vol. 178, Jul 2022.

[4] R. M. French, “Catastrophic forgetting in connectionist networks,” *Trends in Cognitive Sciences*, vol. 3, no. 4, pp. 128–135, 1999.

[5] H. Nosrat, M. Shamsh, S. M. Taheri, and M. H. Sedaghat, “Adaptive networks under non-stationary conditions: Formulation, performance analysis, and application,” *IEEE Trans. on Signal Process.*, vol. 63, pp. 4300–4314, 2015.

[6] J. Chen, C. Richard, and A. H. Sayed, “Diffusion lms over multitask networks,” *IEEE Transactions on Signal Processing*, vol. 63, no. 11, pp. 2733–2748, 2015.

[7] E. Fox, E. B. Sudderth, M. I. Jordan, and A. S. Willsky, “Bayesian nonparametric inference of switching dynamic linear models,” *IEEE Transactions on Signal Processing*, vol. 59, no. 4, pp. 1569–1585, 2011.

[8] J. Ding, S. Shahrampour, K. Heel, and V. Tarokh, “Analysis of multi-state autoregressive models,” *IEEE Transactions on Signal Processing*, vol. 66, no. 9, pp. 2420–2440, 2018.

[9] P. Karimi, M. D. Butala, Z. Zhao, and F. Kamalabdadi, “Efficient model selection in switching linear dynamic systems by graph clustering,” *IEEE Signal Processing Letters*, vol. 29, pp. 2482–2486, 2022.

[10] M. Rabbat and R. Nowak, “Distributed optimization in sensor networks,” *Proc. 3rd Int. Symp. on Inf. Proc. in Sensor Netw.*, pp. 20–27, 2004.

[11] S. Kar and J. M. F. Moura, “Distributed consensus algorithms in sensor networks with imperfect communication: Link failures and channel noise,” *IEEE Trans. on Signal Process.*, vol. 57, pp. 355–369, 2009.

[12] F. Hua, R. Nassif, C. Richard, H. Wang, and A. H. Sayed, “Diffusion LMS with communication delays: Stability and performance analysis,” *IEEE Signal Processing Letters*, vol. 27, pp. 730–734, 2020.

[13] A. H. Sayed, “Adaptation, learning, and optimization over networks,” *Foundations and Trends in Machine Learning*, vol. 7, pp. 311–801, 2014.

[14] J. Chen, Z. J. Towfic, and A. H. Sayed, “Dictionary learning over distributed models,” *IEEE Trans. on Signal Process.*, vol. 63, no. 4, pp. 1001–1016, 2015.

[15] Y. Ye, M. Xiao, and M. Skoglund, “Randomized neural networks based decentralized multi-task learning via hybrid multi-block ADMM,” *IEEE Transactions on Signal Processing*, vol. 69, pp. 2844–2857, 2021.

[16] J. Chen, C. Richard, and A. H. Sayed, “Multitask diffusion adaptation over networks,” *IEEE Trans. on Signal Process.*, vol. 62, no. 16, pp. 4129–4144, 2014.

[17] V. Smith, S. Forte, C. Ma, M. Tákač, M. I. Jordan, and M. Jaggi, “CoCoA: A general framework for communication-efficient distributed optimization,” *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 8590–8638, 2017.

[18] M. Jaggi, V. Smith, M. Tákač, J. Terhorst, S. Krishnan, T. Hofmann, and M. I. Jordan, “Communication-efficient distributed dual coordinate ascent,” *Adv. Neural Inf. Process. Systems*, pp. 3068–3076, 2014.

[19] C. Ma, J. Konečný, M. Jaggi, V. Smith, M. I. Jordan, P. Richtárik, and M. Tákač, “Distributed optimization with arbitrary local solvers,” *Optimization Methods and Software*, vol. 32, no. 4, pp. 813–848, 2017.

[20] L. He, A. Bian, and M. Jaggi, “Cola: Decentralized linear learning,” *Adv. Neural Inf. Process. Syst.*, pp. 4536–4546, 2018.

[21] M. Hellkvist, A. Özçeliklake, and A. Ahlén, “Linear regression with distributed learning: A generalization error perspective,” *IEEE Trans. on Signal Process.*, vol. 69, pp. 5479–5495, 2021.

[22] ——, “Distributed continual learning with CoCoA in high-dimensional linear regression,” *IEEE Transactions on Signal Processing*, vol. 72, pp. 1015–1031, 2024.

[23] D. Yuan, A. Prouvietere, and G. Shi, “Distributed online linear regressions,” *IEEE Trans. on Infor. Theory*, vol. 67, no. 1, pp. 616–639, 2021.

[24] Z. Tu, X. Wang, Y. Hong, L. Wang, D. Yuan, and G. Shi, “Distributed online convex optimization with compressed communication,” in *Adv. in Neural Information Process. Systems*, vol. 35, 2022, pp. 34-493–34-504.

[25] T. Doan, M. Abbana Bennani, B. Mazoure, G. Rabusseau, and P. Alquier, “A theoretical analysis of catastrophic forgetting through the NTK overlap matrix,” ser. Proc. of Machine Learning Research, 2021.