REU 2022: MACHINE LEARNING APPROACHES TO OSCILLATOR AND CLOCK SYNCHRONIZATION CONFIDENTIAL NOTES : NOT FOR DISTRIBUTION

PI : HANBAEK LYU
MENTOR: RICHARD YIM

ABSTRACT. The study of clock synchronization (or coupled oscillators) has been an important subject of research in mathematics and various areas of science for decades, with fruitful applications in many areas including wildfire monitoring, electric power networks, robotic vehicle networks, large-scale information fusion, and wireless sensor networks. However, there has been a gap between our theoretical understanding of systems of coupled oscillators and practical requirements for clock synchronization algorithms in modern application contexts. A past REU project showed that it is possible to use standard machine learning methods to predict long-term behavior of a given coupled oscillator system with surprisingly high accuracy, which often surpasses a baseline prediction based on classical oscillator theory. This REU project aims to further advance the prior machine learning approach to synchronization problems. A major goal is to develop interpretible machine learning methods for synchronization prediction to obtain better insight on the key latent features of the complex dynamical system that are responsible for its long-term behavior.

1. INTRODUCTION

1.1. **Background.** Many important phenomena that we would like to understand – formation of public opinion, trending topics on social networks, growth of stock market, development of cancer cells, and outbreak of epidemics, and collective computation in distributed systems – are closely related to predicting large-scale behavior of networks of locally interacting dynamic agents. Perhaps the most widely studied and mathematically tractable such collective behavior is *synchronization* of couple oscillators (e.g., blinking fireflies, circadian pacemakers, BZ chemical oscillators), which has been a central subject for over 40 years [Str00]. It has found numerous applications in many areas including robotic vehicle networks [NL07], electric power networks [DB12].

The emergence of synchrony from a system of coupled oscillators has a natural and strong connection to the field of distributed algorithms in theoretical computer science. A fundamental building block in designing distributed algorithm is *clock synchronization*, which is to synchronize local clocks on nodes of a network so that a shared notion of time is available across the network. This is crucial especially in modern wireless sensor networks [PSC09], where a large number of low-cost and low-power wireless sensors are deployed over a large region in an ad-hoc manner to collectively perform tasks such as monitoring and measuring large-scale phenomena. Due to the limited resource in each sensor compared to the large size of the network, devising clock synchronization algorithms for wireless sensors poses challenging theoretical questions [SBK05].

For a system of deterministic coupled oscillators (e.g., the Kuramoto model [Kur03]), the entire forward dynamics (i.e., evolution of phase configurations) is analytically determined by 1) the initial phase configuration and 2) the graph structure (see Figure 1). In this project, we are concerned with the fundamental problem of *predicting whether a given system of coupled oscillators will eventually synchronize*

Date: June 30, 2022.

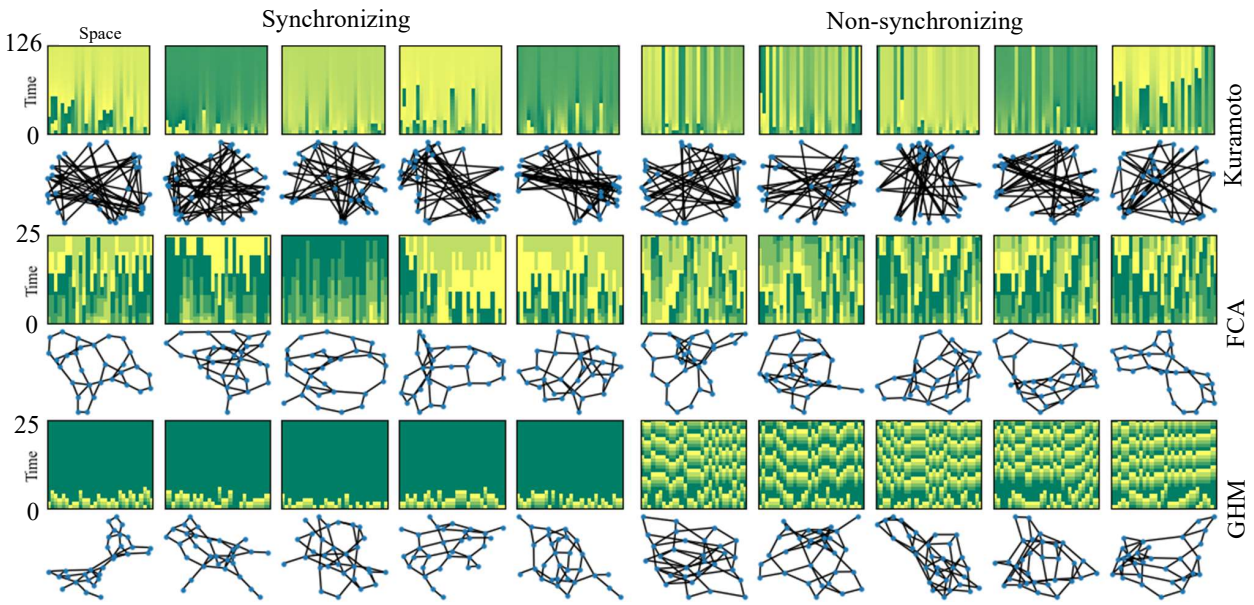

FIGURE 1. Sample points in the 30-node dynamics dataset for synchronization prediction. The heat maps show phase dynamics on graphs beneath them, where colors represent phases and time is measured by iterations from bottom-to-top (e.g. $t = 0$ to $t = 25$). Each example is labeled as 'synchronizing' if it synchronizes at iteration 1758 for the Kuramoto model (70 for FCA and GHM) and 'non-synchronizing' otherwise. Synchronizing examples have mostly uniform color in the top row. For training, only a portion of dynamics is used so that the algorithms rarely see a fully synchronized example.

provided with initial dynamics (first few phase configurations) optionally together with the graph structure. More specifically, we consider the following three types of synchronization prediction problems:

Q1. *Given the initial dynamics and graph structure, can we predict whether the system will eventually synchronize?*

Q2. *Given the initial dynamics and not knowing the graph structure, can we predict whether the system will eventually synchronize?*

Q3. *Given the initial dynamics partially observed on a subset of nodes and possibly not knowing the graph structure, can we predict whether the whole system will eventually synchronize?*

An analytical characterization of synchronization would lead to a perfect algorithm for the synchronization prediction problems above. However, while a number of sufficient conditions on model parameters (e.g., large coupling strength) or on initial configuration (e.g., phase concentration into open half-circle) for synchronization are known, obtaining an analytic or asymptotic solution to the prediction question in general appears to be out of reach, especially when the underlying graphs are heterogeneous and the initial phases are not concentrated. In this work, 'heterogeneous graphs' refers to sets of non-isomorphic simple graphs, where there is one type of node and edge but the connection topology is diverse. Since the global behavior of coupled oscillators is built on non-linear local interactions, as the number of nodes increase and the graphs become more heterogeneous, the behavior of the system becomes rapidly intractable. To provide a sense of the complexity of the problem, note that there are more than 10^9 non-isomorphic connected simple graphs with 11 nodes [McK].

1.2. **Prior work: L2PSync.** In a paper resulted from a past REU project (in 2022 virtually at UCLA) [BYK+20], we take a novel approach that we call learning to predict synchronization (L2PSync), by viewing the synchronization prediction problem as a classification problem for sets of initial dynamics into two classes:

synchronizing or non-synchronizing. While a baseline predictor using concentration principle misses a large proportion of synchronizing examples, standard binary classification algorithms trained on large enough datasets of initial dynamics can successfully predict the unseen future of a system on highly heterogeneous sets of unknown graphs with surprising accuracy. In addition, we find that the full graph information gives only marginal improvements over what we can achieve by only using the initial dynamics.

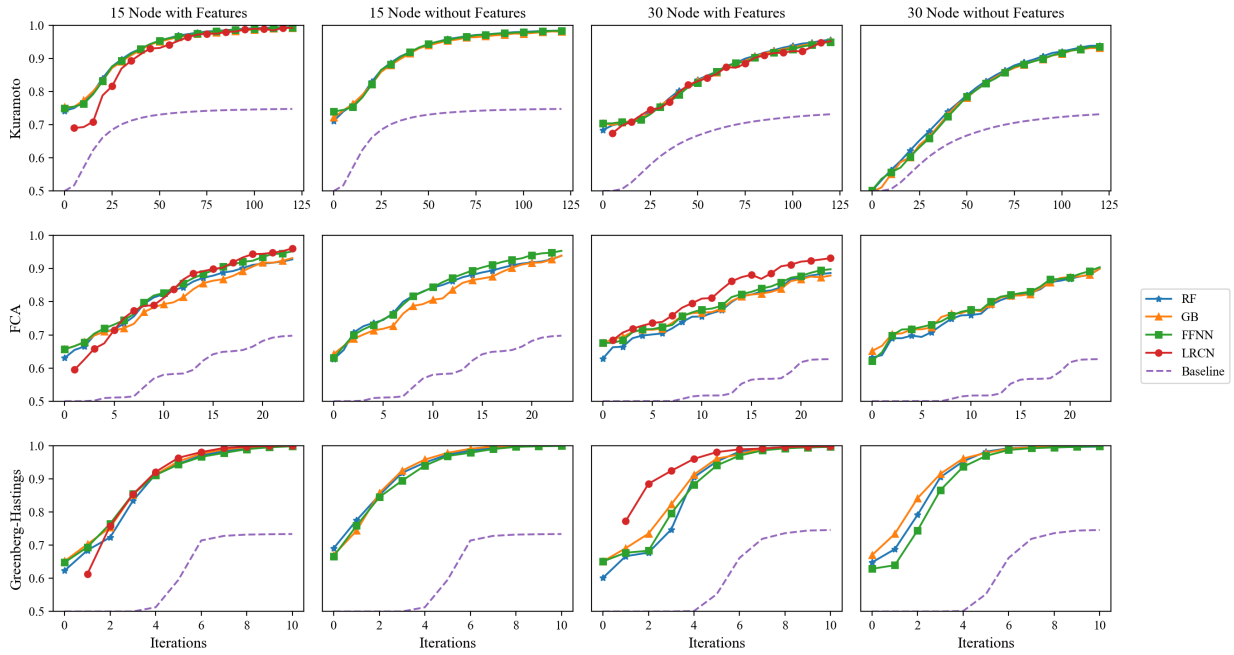

FIGURE 2. Synchronization prediction accuracy of four machine learning algorithms for the KM, FCA and GHM coupled oscillators synchronization. For each of the six datasets in Table **??**, we used 5-fold cross-validation with 80/20 split of test/train. Accuracy is measured by the ratio of the number of correctly classified examples to the total number of examples. Algorithms for the second and the fourth columns are trained only with dynamics up to various iterations indicated on the horizontal axes, whereas the other two columns also use additional graph features.

1.3. **Main problems.** Given a graph $G = (V, E)$, a *phase configuration* is a map $\phi_\bullet : V \to S^1 = \mathbb{R}/\mathbb{Z}$. A time evolution rule for phase configuration $(\phi_\bullet(t))_{t\geq 0}$ is called a *coupling* if it is determined by local phase data, and a *clock synchronization algorithm* if it assumes additional local information. In both schemes, we assume the nodes are indistinguishable.

A fundamental problem in dynamical systems in general is to understand tight interplay between network structure of underlying graph and complex dynamical process defined on the graph. If we concentrate on systems of coupled oscillators, we can formulate this problem precisely as follow.

Problem 1.1 (Characterization)**.** *Fix a coupling and a graph G. Find a necessary and sufficient condition on $(G, \phi_\bullet(0))$ for the phase trajectory $(\phi_\bullet(t))_{t\geq 0}$ on a graph G to synchronize.*

For fixed coupling and underlying graph, we cannot use approximation techniques and hope nonlinear interaction between the oscillators become 'tamed' in some limit: We have to face the nonlinearity directly. Even for the most extensively studied Kuramoto model [ABV+05] on complete graphs, the above problem has not been solved: We only know convergence to equilibrium points [JMB04], properties of

synchronized solution as number of nodes tends to infinity [Kur75], and asymptotic convergence when the coupling strength tends to infinity [HKR16].

On the other hand, we could first fix a (family) of graphs and then ask what type of coupling would synchronize arbitrary phase configuration. This is known as the problem of deriving global synchronization:

Problem 1.2 (Global synchornization)**.** *Fix a family of graphs $\mathcal{G}$. Find a coupling under which arbitrary (or almost all) initial phase configuration on all $G \in \mathcal{G}$ synchronizes.*

The main difficulty in addressing Problems 1.3 and 1.2 is the lack of monotonicity due to cyclic hierarchy of phase space S^1. For some well-known models of pulse-coupled oscillators, Problem 1.2 has been established only when $\mathcal{G}$ is the class of complete graphs or cycles [MS90, KB12, WND13].

One of the main theme in this project is a version of Problem 1.3, which is to understand how standard classification algorithsm used in the framework of L2PSync [BYK+20] were so successful in predicting synchronization of an unknown system. The fact that there are no meaningful difference between the performance of an elementary binary classification algorithm (e.g., random forest) and an advanced deep neural nework architecture (e.g., Graph LRCN) suggests that there might be some interpretible and clear pattern, in the graph topology or short-term dynamcis on them, that these machine learning algorithms are picking up. Thus, we seek to develop machine-learning-based synchronization prediction methods similar to the ones in L2PSync [BYK+20], but that returns interpretible features that the algorithm learns to discriminate synchronizing versus non-synchronizing examples.

Problem 1.3 (Interpretible L2PSync)**.** *Develop a machine learning method that can predict whether a coupled oscillator system will eventually synchronize and also outputs discriminating features that it used for classification of trainign examples.*

1.4. **Our approach: Cellular Automata + Supervised Dictionary Learning.** In this project, we will investigate solutions to the main problems (Problems 1.3 and 1.2) combining two main ingredients of Cellular Automata and a class of machine learning models called *supervised dictionary learning*.

There has not been a significant attempt to study oscillators and clock synchronization problems using Machine Learning, while it has made huge impacts on other fields such as image processing and various classification and clustering tasks. This is because of the fact that most traditional models for oscillator and clock synchronization assume continuous-time and continuous-state for each oscillator so that the dynamics become quickly intractable on heterogeneous underlying graphs. This makes it difficult to analyze the intricate interaction between the network topology and the collective behavior of the oscillators. In order to overcome this difficulty, we *discretize* both the time and the states of the model by using cellular automata in place of the usual continuous models.

Cellular Automata (CA) provide a simple framework for modeling such systems: A vertex coloring $X_t : V \to \mathbb{Z}_\kappa$ on a given graph $G = (V, E)$ updates in discrete time according to a fixed deterministic or random transition rule. CA is a fully discrete model for dynamical systems (discrete-time and discrete-states) that can be defined on arbitrary underlying graphs, which is usually taken to be integer lattices for classical models such as the Game of Life. The main model we will be interested in is called the *Firefly Cellular Automata* (FCA), first introduced by the PI in [Lyu15]. The key idea in this project is to compute all possible (or lots of) FCA trajectories on finite graphs of bounded size, and classify the pair (G, X_0) of underlying graph G and initial κ-coloring into synchronizing and non-synchronizing classes. Note that such an exhaustive computational work is not possible for continuous S^1-valued models such as Kuramoto or Peskin's model. This could help building some intuitions toward addressing the characterization problem (Problem 1.2) for other class of graphs. Moreover, if we build a big enough dataset, then

one could apply recent machine learning techniques such as deep learning or dictionary learning algorithms, in order to recognize some distinguishing features between the two classes, which may otherwise be hard to perceive by human eyes. This could contribute to our understanding of complex dynamical systems in a unique way.

2. FCA: A DISCRETE MODEL FOR OSCILLATOR AND CLOCK SYNCHRONIZATION

In the publications [Lyu15, Lyu16], the PI introduced the κ-color *Firefly Cellular Automata* (FCA) as a discrete model for inhibitory Pulse Coupled Oscillators (PCOs). The intuition is that we view each node as an identical oscillator (firefly) of κ-states, which *blinks* whenever it comes back to a designated *blinking color* $b(\kappa) = \lfloor \frac{\kappa-1}{2} \rfloor$; nodes with post-blinking color $c \in (b(\kappa), \kappa-1]$ in contact with at least one blinking neighbor do not advance, as if their phase update is being inhibited by the blinking ones, and otherwise advance to the next color. (see Figure 3). Namely, a κ-coloring X_t at time t is updated by

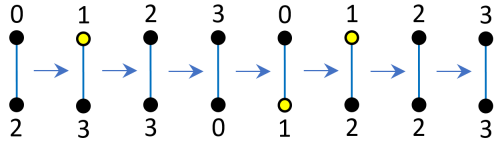

FIGURE 3. 4-color FCA on two connected nodes. $b(4) = 1$ is the blinking color.

$$\text{(FCA)} \quad X_{t+1}(v) = \begin{cases} X_t(v) & \text{if } X_t(v) > b(\kappa) \text{ and } |\{u \in N(v) : X_t(u) = b(\kappa)\}| \geq 1 \\ X_t(v)+1 & \text{otherwise} \end{cases}, \tag{1}$$

where $N(x)$ denotes the set of all neighbors of node x in the underlying graph G. See Figures 3 and 4 for illustration.

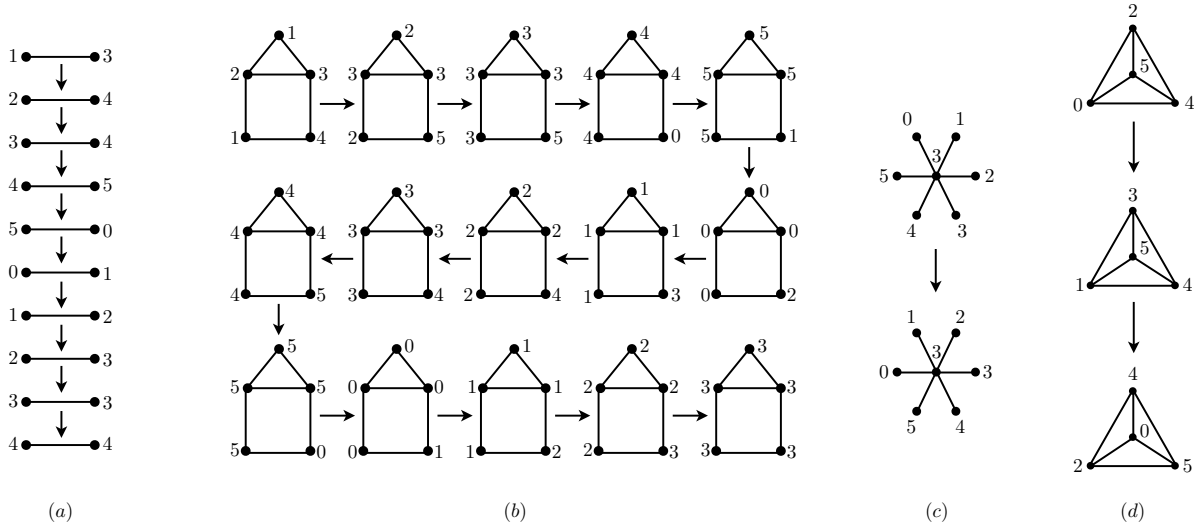

FIGURE 4. Two examples of synchronizing 6-color FCA trajectories are shown in (a) and (b). In (c) and (d), the last configurations are symmetric to the initial ones, so the networks do not synchronize.

Problem 2.1 (Implementing the FCA in python)**.** *Implement the FCA* (1) *in python and generate example dynamics on various underlying graphs and parameter* κ*.*

(i) *The implementation should be able to handle any underlying graph. "networkx" is a popular python package for handling graph objects, but I recommend using my custom package NNetworks as it allows faster neighborhood search for large networks.*

(ii) *Generate example dynamics of FCA for various* κ *on finite paths (see Figure 5).*

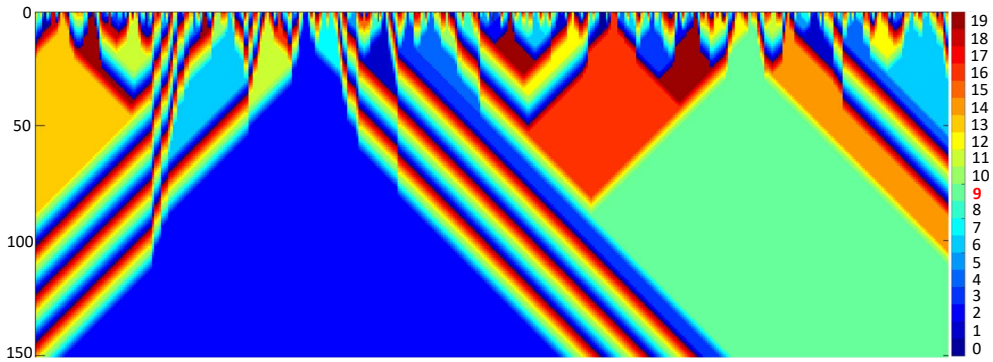

FIGURE 5. Simulation of 20-color FCA on a path of 400 nodes for 150×20. The top rows are a random 20-color initial coloring, and 20 iterations generates the next row, from top to bottom. Various slopes of boundaries indicate particles move in non-constant speeds.

(iii) *Generate example dynamics of FCA for various κ on finite lattices. Make a conjecture for $\kappa = 4$ case on square lattices (see Figure 6).*

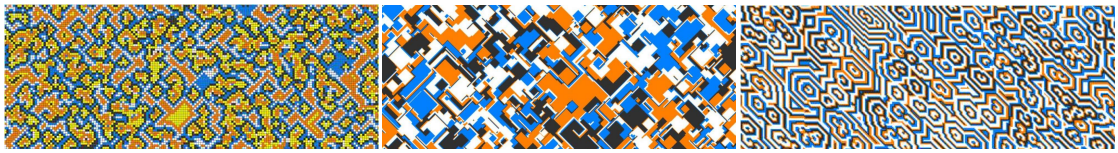

FIGURE 6. 4-color FCA on two connected nodes. $b(4) = 1$ is the blinking color.

Problem 2.2 (Concentration lemma)**.** *Let $G = (V, E)$ be a finite connected graph and let $X_0 : V \to \mathbb{Z}_\kappa = \mathbb{Z}/\kappa\mathbb{Z}$. We say X_0 satisfies the 'half-circle concentration condition' if*

$$\text{All colors used in } X_0 \text{ are confined in } < \kappa/2 \text{ consecutive colors.} \tag{2}$$

The 'concentration lemma' for the FCA [Lyu15, Lem 2.2] *state that the FCA trajectory $(X_t)_{t\geq 0}$ synchronizes if X_0 satisfies the half-circle concentration condition, regardless of the underlying graph G.*

(i) *Verify the concentation lemma for the 4-color FCA using your python implementation. Namely, generate an arbitrary finite connected graph $G = (V, E)$, and let $X_0 : V \to \mathbb{Z}_4$ be such that X_0 uses two consecutive colors (e.g., 0 and 1 or 3 and 0). Simulate the dynamics and verify that the system indeed synchronizes. Make a conjecture on the worst case time to synchronization as a function of the size $|V|$.*

(ii) *Prove the concentration lemma for the 4-color FCA.*

(iii) *Find a counterexample to the concentration lemma for the 4-color FCA.*

A simple application of the above concentration lemma yields the following universal synchronization result for "randomized FCA". Namely, if we assume the oscillators flip an independent coin in order to correctly follow the update rule at each step, this turns the FCA into a Markov chain. Furthermore, a simple application of the above theorem and elementary Markov chain theory shows the following 'stochastic synchronization'.

Theorem 2.3 (Thm 5.1 in [Lyu15])**.** *Let $G = (V, E)$ be a connected graph and fix $\kappa \geq 3$. We give stochasticity to the κ-color FCA on G by assuming either of the followings:*

(i) *(stochastic emission) Each vertex $v \in V$ is present independently at each instant with a fixed probability $p_v \in (0, 1)$;*

(ii) *(stochastic reception) Each edge $e \in E$ is present independently at each instant with a fixed probability $p_e \in (0, 1)$.*

Then the synchronized states constitutes the absorbing states of the resulting Markov chain. In particular, every κ-coloring on G synchronizes with probability 1 in finite expected time.

Problem 2.4. *Generate some examples and verify Theorem 2.3 using your python implementation. Make a conjecture for the expected worst-case synchronization time as a function of the number of nodes. Is it polynomial or exponential? Can such a randomized algorithm be used as a practical clock synchronization algorithm?*

Next, we review some known results on the FCA on finite paths and trees. First, on finite paths of n nodes, it is known that arbitrary initial κ-coloring for all $\kappa \geq 3$ synchronizes in $\Theta(n)$ iterations. In particular, the κ-color FCA solves the global synchronization problem (Problem 1.2) for the class of finite paths.

Theorem 2.5 (Thm 2 in [Lyu15])**.** *For any $\kappa \geq 3$, if $G = (V, E)$ is a path of n vertices and X_0 is any κ-coloring on G, then (G, X_0) synchronizes under FCA dynamics. Moreover, the worst case synchronization time, $T(n)$ satisfies*

$$\kappa\left(\frac{\kappa}{2} - 1 + n\right) \leq T(n) \leq (n-1)\left(\frac{\kappa^2}{2} + 2\kappa - 2\right). \tag{3}$$

Unlike on finite paths, one does not have such a universal synchronization behavior for all $\kappa \geq 3$ on finite trees. Namely, one can easily construct a non-synchronizing κ-coloring on a tree with maximum degree $\geq \kappa$. The obstruction is that if a node v has a post-blinking color $> b(\kappa)$ is facing every single color from $\mathbb{Z}_\kappa$, then it could get delayed constantly so that it may never blink after a finite time. This divides the tree into non-communicating components so synchrony may not emerge (see Figure 4 (c)).

But is the converse of this observation true? Namely, does every κ-coloring on a finite tree T synchronizes if T has maximum degree $< \kappa$? The main results in [Lyu15, Lyu16] establish a phase transition behavior of the κ-color FCA on finite trees with respect to κ and the mximum degree of the tree.

Theorem 2.6. *Let $(X_t)_{t\geq 0}$ be the random κ-color FCA trajectory on a finite tree T with maximum degree Δ.*

(i) *If $\Delta \geq \kappa$, then X_t does not synchronize with positive probability.*
(ii) *For $\kappa \leq 6$, X_t synchronizes with probability 1 iff $\Delta < \kappa$.*
(iii) *For $\kappa \geq 7$, X_t does not synchronize with positive probability on some T with $\Delta \leq \kappa/2 + 1$.*

In fact, the maximum degree condition (i) in the above theorem is a fundamental issue in inhibitory PCOs on finite trees. Namely, nodes with large degree may receive input pulses so often that its phase is constantly inhibited and it may never send pulses to its neighbors. This divides the tree into non-communicating components so global synchrony may not emerge. Also note that, the above result shows that the κ-color FCA solves the global synchronization problem (Problem 1.2) for the class of finite trees with maximum degree $< \kappa$ for $\kappa \in \{3, 4, 5, 6\}$.

Lastly, analyzing local limit cycles using recursive structure of finite trees, Problem 1.3 was addressed for the κ-color FCA on finite trees in the form of the following theorem. In particular, it tells us that the only way the κ-state oscillators do not synchronize on trees is to have at least one oscillator stuck at a post-blinking color. Moreover, this also implies that the κ-color FCA solves the characterization problem (Problem 1.3) for $\kappa \in \{3, 4, 5, 6\}$.

Theorem 2.7. *Let $T = (V, E)$ be any finite tree and let X_0 be any κ-coloring on T for $\kappa \in \{3, 4, 5, 6\}$. Then $(X_t)_{t\geq 0}$ synchronizes if and only if every vertex in T blinks infinitely often in the dynamic.*

Problem 2.8 (Deriving a clock synchronization algorithm)**.** *Let $G = (V, E)$ be a finite connected graph of maximum degree Δ. Fix $\kappa \geq 3$ and a κ-coloring $X_0 : V \to \mathbb{Z}_\kappa$.*

(i) *Write a python code that computes a spanning tree T of G.*

(ii) *Run the κ-color FCA dynamics started from X_0 only using the edges in T. Argue that this will always synchronize if $\Delta < \kappa \leq 6$.*

(iii) *Suppose $\Delta < \kappa \leq 6$. Write a python code that computes a spanning tree of G in a distributed way so that each node updates the list of edges to be used in the final spanning tree. Combine this with the κ-color FCA to make a clock synchronization algorithm. Argue that such a clock synchronization algorithm synchronizes arbitrary initial κ-coloring regardless of G as long as $\Delta < \kappa \leq 6$.*

3. Applying ML classification algorithms to FCA database

Classification is a classical problem in machine learning, where the input is a labeled dataset and the goal is to build an algorithm that predicts the unknown label of an input data (e.g., classification for the MNIST dataset of handwritten digits). There are classical methods such as the Support Vector Machine (SVM), logistic regression, Naive Bayes classifier, as well as more modern ones based on deep neural networks.

Deep Learning is an exploding area of machine learning based on data representations using multiple levels of abstraction. The resulting processing structure is a layered graph (or *neural network*) whose nodes correspond to various types of linear or non-linear transformations. These neural networks can be used for large-scale data problems like classification, feature detection, and topic modeling, all of which are important tools for biomedical big data analysis. Deep neural network algorithms have recently obtained state of the art results for classification of large datasets due to advancement in computing power, availability of massive amounts of observed data, and the development of new techniques.

Below we present some approaches of using Machine Learning classification algorithms on the clock synchronization problem modeled by the FCA.

Problem 3.1 (FCA database generation)**.** *Fix $\kappa \geq 3$ and $n \geq 1$.*

(i) *For small n (say, $n = 10$), generate a database of triples*

$$(G, X_0, \mathbf{1}(X_t \text{ synchronizes})) \tag{4}$$

where $G = (V, E)$ runs over all finite connected graphs of $\leq n$ nodes, X_0 runs over all κ-colorings of G, and $(X_t)_{t \geq 1}$ is the κ-color trajectory on G started at X_0.

(ii) *For moderate n (say, $n = 100$), generate a similar database as in (i) by randomly generating the underlying graph G and the initial coloring X_0.*

Problem 3.2 (FCA synchronization predictor)**.** *Use the database in 3.1 to build a predictor $\mathscr{F}$ that takes the pair (G, X_0) as an input and predicts the indicator $\mathbf{1}(X_t \text{ synchronizes})$.*

(i) *Build such a FCA predictor $\mathscr{F}$ by training standard classification algorithms (e.g., SVM, Naive Bayes, Neural Networks) on the database in 3.1.*

(ii) *Evaluate the performance of your predictor on some randomly generated test dynamics. Plot the result with respect to the size of the graph.*

(iii) *In the training step in (ii), use the known theoretical results as a warm start. Does this improve the performance of your predictor?*

4. Applying ML dictionary learning algorithms to FCA database

Some drawbacks to the approaches using deep neural network include that they are not always convergent, are not well understood mathematically, and the output classifications can randomly fail without warning. An alternative approach for dealing with massive datasets is low-rank matrix factorizations

[LS99] and topic modeling [BNJ03]. Topic models have been shown to efficiently capture latent intrinsic structures of text data in natural language processing tasks [SG07, BCD10].

Topic models combine data modeling with optimization to learn interpretable and consistent feature structures in data. We propose a novel approach that combines the interpretability and predictability of topic modeling learned representations with some of the attributes of deep neural networks, introducing a deep non-negative matrix factorization (NMF) framework capable of producing reliable, interpretable, and predictable hierarchical classification of large-scale data, far exceeding existing approaches.

In this section, we give a brief overview on topic modeling methods based on matrix and tensor factorization algorithms.

4.1. **Dictionary Learning.** *Dictionary-learning* algorithms are machine-learning techniques that are able to learn interpretable latent structures of complex data sets and are applied regularly in the data analysis of text and images. Such algorithms usually consist of two steps. First, one samples a large number of structured subsets of a data set (e.g., square patches of an image or collections of a few sentences of a text); Second, one finds a set of basis elements such that taking a *nonnegative* linear combination of them can successfully approximate each of the sampled mesoscale patches. Such a set of basis elements is called a *dictionary*, and one can interpret each basis element as a latent structure of the data set. As an example, consider the image of the artwork CYCLE by M. C. Escher in Figure 7**a**. We first sample 10,000 square patches of 21×21 pixels, and we then use a nonnegative matrix factorization (NMF) [LS99] algorithm to find a dictionary with $r = 25$ square patches (see Figure 7**a**). Each element of the learned dictionary describes a latent shape in the image.

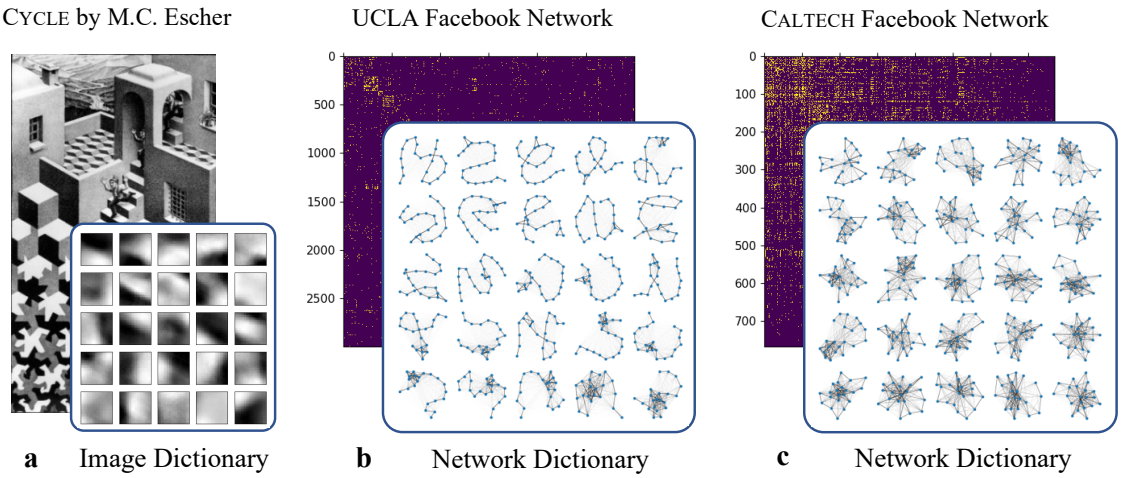

FIGURE 7. Illustration of matrix factorization. Each column of the data matrix is approximated by the linear combination of the columns of the dictionary matrix with coefficients given by the corresponding column of the code matrix. Figure excerpted from [LK+20].

4.2. **Matrix Factorization.** *Matrix factorization* is one of the fundamental tools in dictionary learning problems. Given a large data matrix $\mathbf{X}$, can we find some small number of "dictionary vectors" so that we can represent each column of the data matrix as a linear combination of dictionary vectors? More precisely, given a data matrix $\mathbf{X} \in \mathbb{R}^{d\times n}$ and a rank parameter $r \in \mathbb{N}$, we wish to factorize X into the product of $\mathbf{W} \in \mathbb{R}^{d\times r}$ and $\mathbf{H} \in \mathbb{R}^{r\times n}$ by solving the following optimization problem

$$\text{(Matrix Factorization)} \qquad \inf_{\mathbf{W}\in\mathbb{R}^{d\times r},\mathbf{H}\in\mathbb{R}^{r\times n}} \|\mathbf{X}-\mathbf{W}\mathbf{H}\|_F^2. \tag{5}$$

Here $\mathbf{W}$ is called the *dictionary* and $\mathbf{H}$ is the *code* of data $\mathbf{X}$ using dictionary $\mathbf{W}$. A solution of such matrix factorization problem is illustrated in Figure 8.

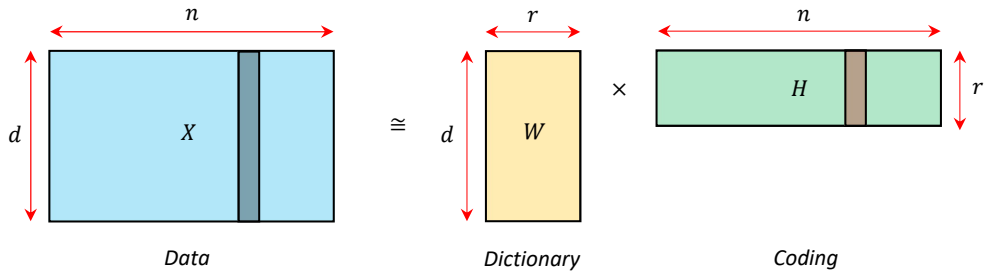

FIGURE 8. Illustration of matrix factorization. Each column of the data matrix is approximated by the linear combination of the columns of the dictionary matrix with coefficients given by the corresponding column of the code matrix. Figure excerpted from [LNB20].

Recall the method of classical least squares:

$$\hat{\mathbf{H}} = \underset{\mathbf{H}\in\mathbb{R}^{r\times n}}{\operatorname{argmin}} \|\underbrace{\mathbf{X}}_{\text{given}} - \underbrace{\mathbf{W}}_{\text{given}}\underbrace{\mathbf{H}}_{\text{TBD}}\|_F^2 + \lambda\|\underbrace{\mathbf{H}}_{\text{TBD}}\|_F^2. \tag{6}$$

One can think of (6) as a matrix factorization problem where the first factor (dictionary) is known and we are only looking for the code $\mathbf{H}$. Becase of this reason, we also call (6) as a *coding* problem, which is also called a *sparse coding* problem if the regularizer λ is positive[1].

While the optimization problem (5) is an *unconstrained* problem in the sense that we are free to choose any dictionary $\mathbf{W}\in\mathbb{R}^{d\times r}$ and code $\mathbf{H}\in\mathbb{R}^{r\times n}$, We will be more interested in its *constrained* versions, in particular using nonnegativity constraints. Consider the following modifications:

$$\text{(Dictionary Learning)} \qquad \inf_{\mathbf{W}\in\mathbb{R}^{d\times r},\mathbf{H}\in\mathbb{R}^{r\times n}_{\geq 0}} \|\mathbf{X}-\mathbf{W}\mathbf{H}\|_F^2, \tag{7}$$

$$\text{(NMF)} \qquad \inf_{\mathbf{W}\in\mathbb{R}^{d\times r}_{\geq 0},\mathbf{H}\in\mathbb{R}^{r\times n}_{\geq 0}} \|\mathbf{X}-\mathbf{W}\mathbf{H}\|_F^2. \tag{8}$$

In (7), we imposed an additional constraint that the code $\mathbf{H}\in\mathbb{R}^{r\times n}$ should only take nonngative entries. This allows us to interpret the columns of dictionary $\mathbf{W}$ as 'features' that directly contribute to composing the columns of $\mathbf{X}$ (samples)[2]. We can go even further and also impose the same nonnegativity constraint on the dictioanry matrix $\mathbf{W}\in\mathbb{R}^{r\times n}$ as in (8). This problem is called *nonnegative matrix factorization* (NMF), which is an extremely popular technique in modern machine learning for its high interpretibility and mathematical simplicity.

In NMF, each column of the data matrix has to be represented as a non-negative linear combination of dictionaries. Hence the dictionaries must be "positive parts" of the columns of the data matrix. When each column consists of a human face image, NMF learns the parts of human face (e.g., eyes, nodes, mouth). This is in contrast to principal component analysis and vector quantization: Due to cancellation between eigenvectors, each 'eigenface' does not have to be parts of face. NMF was popularized by Lee and Seung in their Nature paper in 1999.

[1] It is more effective to use an L_1-regularizer $\lambda\|\mathbf{H}\|_1$ for the purpose of sparse coding. This is also closely related to (Lagrangian version of) the popular algorithm of LASSO in statistics.

[2] A general formulation of dictionary learning may also impose a convex constraint $\mathscr{C}\subseteq\mathbb{R}^{d\times r}$ for dictionary $\mathbf{W}$ along with the nonnegativity constraint on the code $\mathbf{H}$.

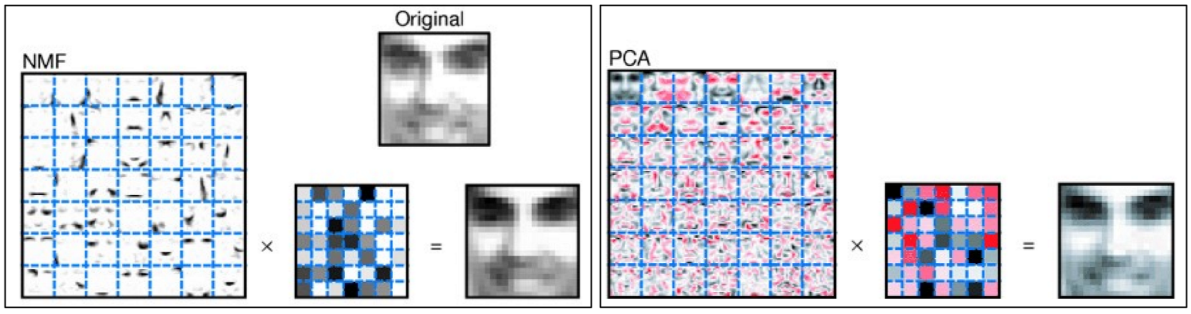

FIGURE 9. Learning parts of human faces by NMF (exerpted from [LS01])

4.3. **Applications in Image processing.** Before we discuss how to solve matrix factorization problems, we give some examples of matrix factorization in the context of image processing through image dictionary learning.

Suppose we are given an image, and we are intrested in the following *unsupervised problem* of learning 'essential features in all of its $k \times k$ patches'. In other words, we would like to know what important features are there that if we cut out a large number of $k \times k$ square patches from the original image. Of course, still this problem is not precisely defined. Even though we can state a precise optimization problem for it, let's instead describe the algorithm first and then see what exactly we are solving.

As we have briefly mentioned in Section (4.1), for dictionary learning we first do patch sampling and then feature extraction. For patch sampling, we get a large number N (e.g., N =10K) of $k \times k$ (e.g., 21×21) square patches from the original image. We will vectorize each such patch as a k^2 dimensional column vector and form a data matrix $\mathbf{X}$ of shape $k^2 \times N$. We then use an algorithm[3] for NMF to get a dictionary matrix $\mathbf{W} \in \mathbb{R}^{k^2 \times r}$ and code matrix $\mathbf{H} \in \mathbb{R}^{r \times N}$ with $r = 25$. Each of the 25 columns of $\mathbf{W}$ are vectorized versions of the extracted features or 'basis shapes'. Reshaping these k^2 dimensional column vectors into $k \times k$ square form, we obtain the image dictionary of size 25, as shown in Figure 10.

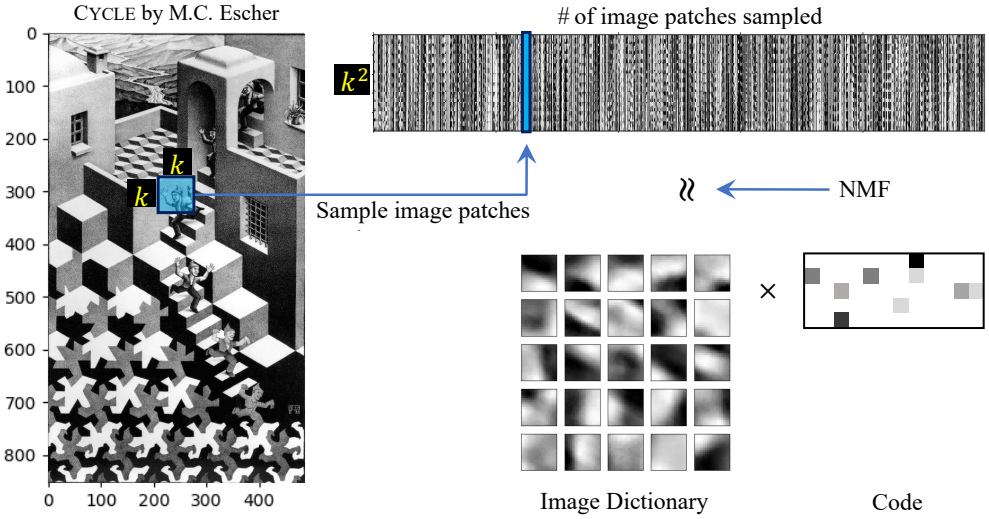

FIGURE 10. Examples of image dictionary learning by patch sampling and NMF.

After we learn a set of dictionary images from a given image, we can use them to reconstruct the original image. Moreover, if the given image is corrupted with some noise, then this reconstruction using learned image dictionary can be used for image denoising. Figure 11 gives an example of image dictionary learning and image denoising with noise separation. We omit details here.

[3]We will discuss algorithms for solving matrix factorization problems later

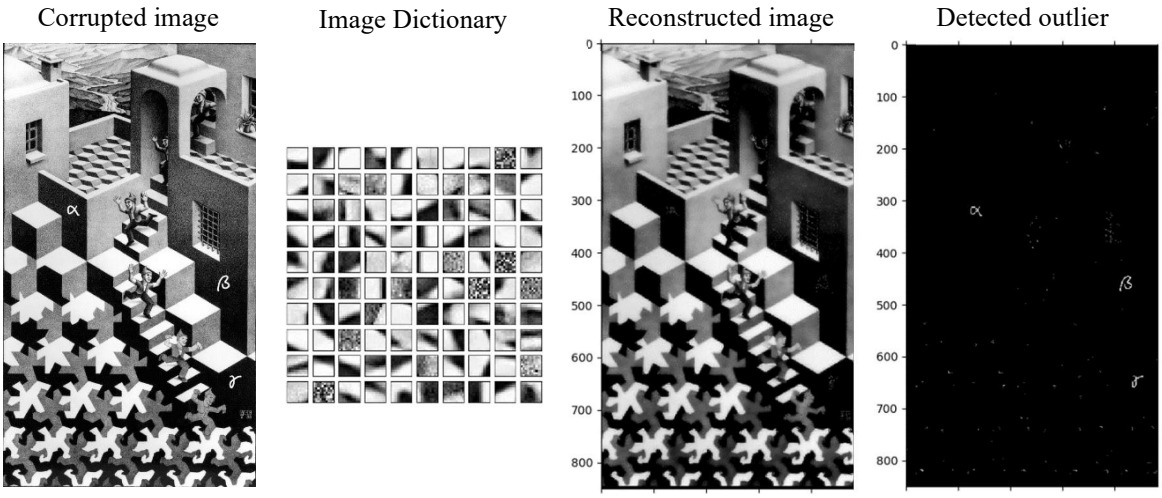

FIGURE 11. Examples of image dictionary learning by patch sampling and NMF. The learned image dictionary is used for denoising the original image corrupted with noises 'α', 'β', and 'γ'.

Problem 4.1 (NMF on the FCA database)**.** *Fix $\kappa \geq 3$ and $n \geq 1$.*

(i) *Generate a database of N **non-synchronizing** pairs (G, X_0), where G is a connected graph of n nodes, X_0 is an initial κ-coloring on G, and the corresponding trajectory X_t do **not** synchronize.*

(ii) *Let $M = \binom{n}{2} + n$ and form a $M \times N$ data matrix X using the database in (i), where the first $\binom{n}{2}$ rows correspond to the list of edges in G and the rest n rows correspond to the initial coloring X_0. Apply Nonnegative Matrix Factorization algorithms to obtain an approximate factorization $X \approx WH$. The columns of the dictionary matrix W will give features of non-sychronizing FCA pairs (G, X_0). Can you interprete these features?*

(iii) *Do a similar work for a databse of synchronizing pairs.*

4.4. **Supervised dictionary learning.** There has been extensive research on making dictionary learning models adapted to also perform classification tasks by supervising the dictionary learning process using additional class labels. Note that dictionary learning and classification are not necessarily aligned objectives, so some degree of trade-off is necessary when one seeks to achieve both goals at the same time. *Supervised dictionary learning* (SDL) provides systematic approaches for such a multi-objective task. The general framework of SDL was introduced in [MPS+08]. A stochastic formulation of SDL was proposed as 'task-driven dictionary learning' in [MBP11]. A similar SDL-type framework of discriminative K-SVD was proposed for face recognition [ZL10]. SDL has also found numerous applications in various other problem domains including speech and emotion recognition [GFG+14], music genre classification [ZHL+15], concurrent brain network inference [ZHL+15], and structure-aware clustering [YE17]. More recently, supervised variants of NMF, as well as PCA, were proposed in [AAG18, LSF+19, RBK+20]. See also the survey [GFGK15] on SDL.

Suppose we are given with n labeled signals $(\boldsymbol{x}_i, y_i)$ for $i = 1, \ldots, n$, where $\boldsymbol{x}_i \in \mathbb{R}^p$ is a p-dimensional signal and $y_i \in \{0, 1\}$ is its binary label. In the classical *dictionary learning* (DL) literature [MBPS10, Mai13a, Mai13b], one seeks to find a dictionary $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_r] \in \mathbb{R}^{p \times r}$ ($r \ll p$) that is *reconstructive* in the sense that the observed signals $\boldsymbol{x}_i$ can be effectively reconstructed as (or approximated by) the linear transform $\mathbf{W}\mathbf{h}_i$ of the 'atoms' $\mathbf{w}_1, \ldots, \mathbf{w}_r \in \mathbb{R}^p$ for some suitable (sparse) 'code' $\mathbf{h}_i \in \mathbb{R}^r$. However, the best reconstructive dictionary $\mathbf{W}$ may not be very effective for the classification tasks. In the *supervised dictionary learning* (SDL) literature [MPS+08], one desires dictionary that is reconstructive as well as *discriminative* in that such a compressed representation of signals is adapted to predicting the class labels y_i.

More precisely, consider the following probability distribution $\mathbf{g}(\mathbf{a}) = (g_0(\mathbf{a}), g_1(\mathbf{a}))$ on $\{0,1\}$ with *activation* $\mathbf{a} = (a_1, \ldots, a_\kappa)$ given by

$$g_0(\mathbf{a}) = \frac{1}{1+\sum_{c=1}^{\kappa} h(a_c)}, \qquad g_1(\mathbf{a}) = \frac{h(a_1)}{1+\sum_{c=1}^{\kappa} h(a_c)}, \tag{9}$$

where $h : \mathbb{R} \to [0,\infty)$ is a fixed *score function*. For instance, taking $h(\cdot) = \exp(\cdot)$ results in the classical logistic regression. We then model the given training data $(\boldsymbol{x}_i, y_i)$ as

$$\boldsymbol{x}_i = \mathbf{W}\mathbf{h}_i \quad \text{and} \quad y_i \,|\, \boldsymbol{x}_i \sim \text{Multinomial}(1, \mathbf{g}(\mathbf{a}_i)) \quad \text{for } i = 1, \ldots, n, \tag{10}$$

where we allow the activation $\mathbf{a}_i$ to depend on the signal $\boldsymbol{x}_i$, latent factors $\mathbf{W}$ and $\mathbf{h}_i$, and an additional model parameter $\boldsymbol{\beta}$ through some functional relation.

As we seek to balance the tasks of dictionary learning and classification, the objective of SDL can naturally be formulated as a multi-objective optimization problem as below:

$$\min_{\mathbf{W},\mathbf{H},\boldsymbol{\beta}} \quad L(\mathbf{W},\mathbf{H},\boldsymbol{\beta}) := \left(\sum_{i=1}^{n} \ell(y_i, \mathbf{g}(\mathbf{a}(\boldsymbol{x}_i, \mathbf{W}, \mathbf{h}_i, \boldsymbol{\beta}))) \right) + \xi \|\mathbf{X}_{\text{data}} - \mathbf{W}\mathbf{H}\|_F^2 \tag{11}$$

$$\text{subject to:} \quad \text{Constraints on } \mathbf{W} \in \mathbb{R}^{p\times r}, \, \mathbf{H} \in \mathbb{R}^{r\times n}, \text{ and } \boldsymbol{\beta} \in \mathbb{R}^{r\times \kappa} \tag{12}$$

where $\mathbf{X}_{\text{data}} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{p\times n}$, $\mathbf{H} = [\mathbf{h}_1, \ldots, \mathbf{h}_n] \in \mathbb{R}^{r\times n}$, and $\ell(\cdot)$ is a classification loss and is usually taken as the negative log likelihood

$$\ell(y_i, \mathbf{g}(\mathbf{a}(\boldsymbol{x}_i, \mathbf{W}, \mathbf{h}_i, \boldsymbol{\beta}))) := -\sum_{j=0}^{1} \mathbf{1}(y_i = j) \log\left\{ g_j(\mathbf{a}(\boldsymbol{x}_i, \mathbf{W}, \mathbf{h}_i, \boldsymbol{\beta})) \right\}. \tag{13}$$

Here, the *tuning parameter* ξ controls the trade-off between the two objectives of classification and dictionary learning. We allow to put desired constraints on the parameters $\{\mathbf{W}, \mathbf{H}, \boldsymbol{\beta}\}$. In particular, we will consider nonnegativity constraints on $\mathbf{W}$ and $\mathbf{H}$ as in the supervised nonnegative matrix factorization (SNMF) model [AAG18, LSF+19] to enjoy the nice interpretability of NMF in the supervised setting.

Various models of the form (11) have been proposed in the past two decades. We divide them into two categories, depending on whether the classification model $\mathbf{g}$ is either 'feature-based' or 'filter-based'. The classification function $\mathbf{g}$ in *feature-based SDL* (SDL-feat) makes use of the code $\mathbf{h}_i$, which is a r-dimensional feature of $\boldsymbol{x}_i$ extracted by the dictionary $\mathbf{W}$. On the other hand, *filter-based SDL* (SDL-filt) uses the r-dimensional filtered input $\mathbf{W}^T\boldsymbol{x}_i$ instead of the code $\mathbf{h}_i$ in the classification/prediction step. In this work, we consider the following two types of multinomial prediction models:

(SDL-feat) Feature-based classification: $\mathbf{a}(\boldsymbol{x}_i, \mathbf{W}, \mathbf{h}_i, \boldsymbol{\beta}) = \boldsymbol{\beta}^T\mathbf{h}_i$;

(SDL-filt) Filter-based classification: $\mathbf{a}(\boldsymbol{x}_i, \mathbf{W}, \mathbf{h}_i, \boldsymbol{\beta}) = \boldsymbol{\beta}^T\mathbf{W}^T\boldsymbol{x}_i$.

Feature-based models include the classical ones by Mairal et al. (see, e.g., [MPS+08, MBP11]) as well as the more recent model of Convolutional Matrix Factorization by Kim et al. [KPO+16] for a contextual text recommendation system. One of the downsides of SDL-feat for classification tasks is that for a new test signal $\boldsymbol{x}$, its correct code representation $\mathbf{h}$ may need to be learned in a supervised fashion by using an unknown true label y of $\boldsymbol{x}$. Since y can assume $\kappa+1$ different class labels, one can solve κ instances of 'supervised sparse coding' to make a prediction for test signals.

Below in Figures 12, 13, and 14, we give two illustrative examples of SDL applications on 1) fake job posting detection by supervised topic modeling (using SDL-filt) and 2) pneumonia detection by supervised image dictionary learning for chest x-ray images (using SDL-feat). See [LLY22] for more details.

Problem 4.2 (SDL on the FCA database)**.** *Fix* $\kappa \geq 3$ *and* $n \geq 1$.

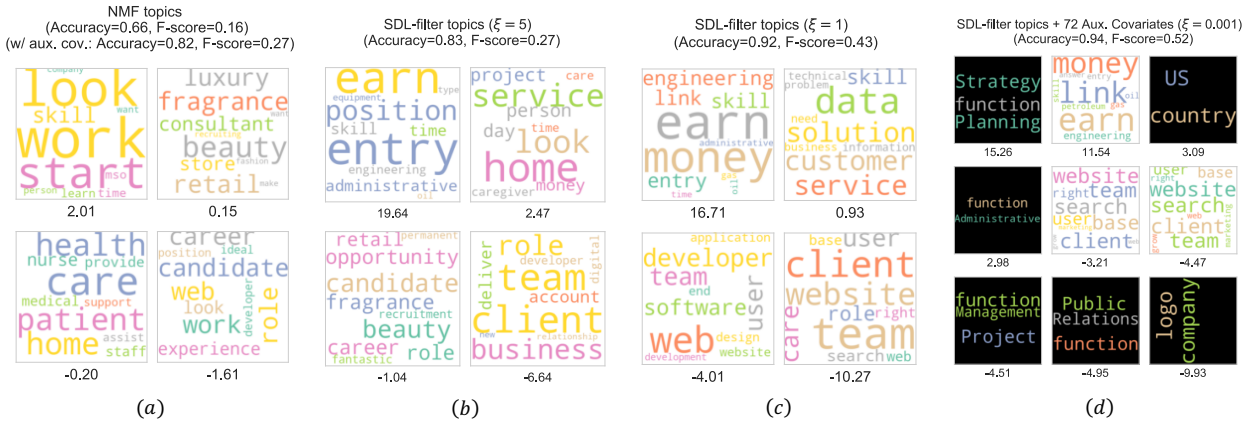

FIGURE 12. Comparison between (supervised) topics learned by NMF and SDL-filter for the fake job posting data. (a) Four out of 25 topics learned by NMF are shown together with the corresponding logistic regression coefficients. (b) Four out of 20 supervised topics learned by SDL-filter with tuning parameter $\xi = 5$ are shown together with the corresponding logistic regression coefficients. (c) Similar as (b) but with tuning parameter $\xi = 1$. (d) Nine out of 20 supervised topics (white background)+ 72 auxiliary covariates (dark background) learned from SDL-filt with 72 auxiliary covariates are shown with their corresponding logistic regression coefficients. Corresponding classification accuracy and F-scores are also shown in the subtitles with fake job postings being the positive examples. For topic wordclouds (white background), word sizes are proportional to their frequency. Figure excerpted from [LLY22].

FIGURE 13. The normal chest X-ray (left panel) depicts clear lungs without any areas of abnormal opacification in the image. Bacterial pneumonia (middle) typically exhibits a focal lobar consolidation, in this case in the right upper lobe (white arrows), whereas viral pneumonia (right) manifests with a more diffuse interstitial pattern in both lungs. The figure and the description are excerpted from [KGC+18].

(i) *Generate a database of N synchronizing and another N non-synchronizing pairs (G, X_0), where G is a connected graph of n nodes, X_0 is an initial κ-coloring on G, and the corresponding trajectory X_t do* ***not*** *synchronize.*

(ii) *Let $M = \binom{n}{2} + n$ and form a $M \times 2N$ data matrix X using the database in (i), where the first $\binom{n}{2}$ rows correspond to the list of edges in G and the rest n rows correspond to the initial coloring X_0. Correspondingly, let Y denote $1 \times N$ row vector of class labels (e.g., '1' for synchronizing and '0' for non-synchronizing). Apply Supervised Dictionary Learning (e.g., SDL-filt) to learn classi-discriminating dictionaries (as in Figures 12 and 14). Can you interprete these features?*

(iii) *As a targeted instance, have all N synchronizing examples to be initially satisfy half-circle concentration. The supervised dictionary should pick up half-circle concentration as a discriminating feature. Does SDL actually re-discover this theoretical result?*

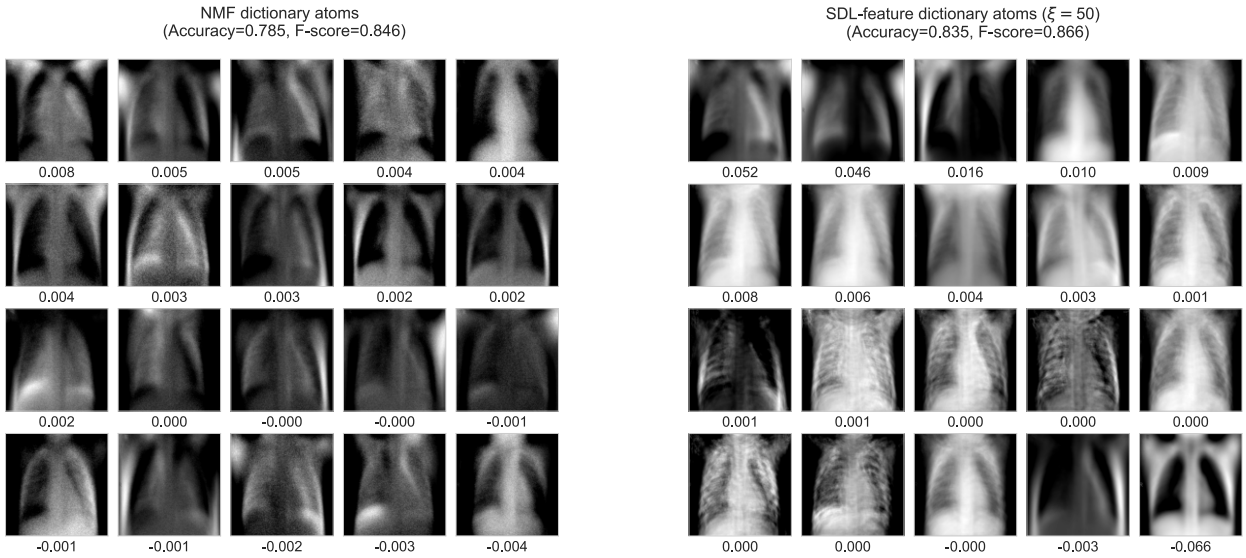

FIGURE 14. 25 dictionary atoms learned from chest X-ray images by NMF and SDL-feature with $\xi = 50$. Corresponding logistic regression coefficients, as well as classification performances, are also shown. For SDL-feature, we used L_1 regularization coefficient of 5 for the code matrix $\mathbf{H}$ and no L_2-regularization. Positive regression coefficients indicate a positive correlation with having pneumonia. There is a clear contrast between the two extreme atoms (upper left and lower right) according to their correlation with pneumonia.

4.5. **Dictionary learning by tensor factorization.** [To be added] ***

5. Project steps and guidelines

We suggest the following steps and milestones for the project, but these are subject to change based on group progress, findings, and timeline.

5.1. **Project steps.**

(1) **Intro.** Read over this writeup. Understand how FCA works on various graphs. Also, get to know how various dictionary learning and matrix factorization models work.
   [1] Slides for the FCA talk at U. Aberta (link)
   [2] Slides for "Matrix and Tensor Factorization Models: Applications, Algorithms, and Theory" (link)

(2) **Preliminary Writing.** You will log all your work in the "braindump.tex" file, which is in your Dropbox. Overleaf is also a good way to organize your notes with the team.

(3) **Preliminary Reading.** Read relevant literature (start with the papers referenced in this writeup). Note: when reading research papers, it is not necessary to understand every detail in the paper. You want to get an overall understanding of the main ideas. As you read the papers, add their citations and brief summaries to the braindump.tex file. This will build your annotated bibliography (see due date below). Note that these are just a start. An important part of research is to learn how to do a literature search. You will need to find and read other related papers as well (try Google Scholar, MathSciNet, Arxiv, etc.).

(4) **Useful github repositories.** Read through the READMEs and examples of the following repositories
   [1] L2PSync repository: (link)
   [2] Supervised Dictionary Learning repository: (link)
   [3] UCLA Machine Learning course repository (link) (*Specifically, run the matrix factorization jupyter notebook.*)
   [4] Online Nonnegative Matrix Factorization algorithm (link)
   [5] Online CP Dictionary Learning algorithm (link)
   [6] NNetwork (neighborhood-based python class for graphs and networks) (link)
   [7] Motif sampling algorithms (link)

(5) **NMF and topic modeling implementation using SKlearn.** Run through the following tutorials and documentations:
   [1] Basic NMF using Nimfa (link).
   [2] Topic modeling using NMF in SKlearn (link).
   [3] Image reconstruction using NMF in SKlearn (link)

(6) **Implementing the FCA in python and generating examples:** Problem 2.1 (*Can use the previous one in L2PSync*)

(7) **Experimental validation of the concentration lemma and randomized FCA synchronization:** Problems 2.2 and 2.4 (*Use SDL to see if the algorithm can pick up half-circle concentration.*)

(8) **FCA database generation:** Problem 3.1 (*Can use the previous one in L2PSync*)

(9) **Building FCA synchronization predictor:** Problem 3.2 (*Can refer to the previous one in L2PSync*)

(10) **Applying NMF on the FCA database:** Problem 4.1.

(11) **Applying Superviesd Dictionary Learning on the FCA database:**

(12) **Applying Online CP-Dictionary Learning on the FCA database:**

End Goal: We will understand how the FCA, as a fully discrete model for oscillator and clock synchronization, works on various graphs. We will generate database of FCA trajectories on various initial

data (Graph, Coloring) with synchronization labels. We will apply some standard classification algorithms to this database to build synchronization predictor, and also matrix factorization based dictionary learning algorithms to extract key features of synchronizing or non-synchronizing pairs of initial data. Furthermore, we will apply supervised dictionary learning methods to learn interpretible classification propocols.

5.2. **Important dates.** Please note the following important dates.

July 1 Start of REU (Orientation 10am at VV303)
July 4 July 4th Holiday
July 8 Annotated bibliography due
July 27 Practice midterm presentations
July 29 Midterm presentations
Aug 3 Practice final presentations
Aug 29 Practice Final presentations
Aug 31 Final day of REU. Final report and code due by 5pm.

5.3. **Guidelines and tips. Office locations and contacts:**

- Dr. Hanbaek Lyu, `hlyu@math.wisc.edu` (Van Vleck 303)
- Richard Yim, `richyim555@ucla.edu` (At UC Davis, Virtual)

Please follow these guidelines throughout the project:

- Work should begin daily at 9am sharp and end no earlier than 5pm. Sick days or deviation from this schedule should be brought to the PI's attention immediately.
- Please feel free to contact mentors and PIs with any personal or group dynamic concerns. We are here to support you. There are also professional services available for counseling, please see your welcome folder.
- Every student comes with a different level of experience and background, and there are no demands or expectations that everyone brings the same level of expertise. However, everyone can still contribute the same amount, and we are here to help you find tasks that are suited to your strengths. This should be a learning experience for everyone!
- Questions brought to the mentors should be discussed as a group. Please take advantage of the various expertise among your group members first!
- Please come prepared to your regular meetings. For efficiency, ahead of time you should organize with your group an agenda of topics and questions to be discussed. It may also be necessary to open and organize documents, figures, and code before your meeting!
- Every student must speak *English* while discussing anything related to the group or project. In addition, all students are expected to perform all the work.
- Never store raw data on your computer. Always use a strong password to lock your computer as well as any data files (raw or coded).
- When writing in the notes, slides, or final report, use the pronoun "we." This is a group project, and the group takes ownership of all its members work. All authors will be listed alphabetically on any published works that come out of this project, unless the mentors deem special circumstances that suggest otherwise.
- Keep a list of agreed upon notation at the beginning of the braindump and be consistent with usage throughout the document.
- When including such images in latex, please always include a comment in the tex file with instructions for where the image files are stored as well as what code was used to generate the image (and where that code is stored).

- Please use bibtex for all bibliographies! You can easily find bibtex sources for papers by e.g. using Google Scholar (go to your Settings in Google Scholar and select to view citations in bibtex – then you can simply click on quotation marks next to your search results to view the bibtex code).
- In Latex, please never hard code references to sections, figures, equations, etc. but instead always use "ref" or "eqref".
- Document all your code! Please include comments in your code so that others can read and use it.

REFERENCES

[AAG18] Woody Austin, Dylan Anderson, and Joydeep Ghosh, *Fully supervised non-negative matrix factorization for feature extraction*, IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium, IEEE, 2018, pp. 5772–5775.

[ABV+05] Juan A Acebrón, Luis L Bonilla, Conrad J Pérez Vicente, Félix Ritort, and Renato Spigler, *The kuramoto model: A simple paradigm for synchronization phenomena*, Reviews of modern physics **77** (2005), no. 1, 137.

[BCD10] David Blei, Lawrence Carin, and David Dunson, *Probabilistic topic models: A focus on graphical model design and applications to document and image analysis*, IEEE signal processing magazine **27** (2010), no. 6, 55.

[BNJ03] David M Blei, Andrew Y Ng, and Michael I Jordan, *Latent dirichlet allocation*, Journal of machine Learning research **3** (2003), no. Jan, 993–1022.

[BYK+20] Hardeep Bassi, Richard Yim, Rohith Kodukula, Joshua Vendrow, Cherlin Zhu, and Hanbaek Lyu, *Learning to predict synchronization of coupled oscillators on heterogeneous graphs*, arXiv preprint arXiv:2012.14048 (2020).

[DB12] Florian Dorfler and Francesco Bullo, *Synchronization and transient stability in power networks and nonuniform kuramoto oscillators*, SIAM Journal on Control and Optimization **50** (2012), no. 3, 1616–1642.

[GFG+14] Mehrdad J Gangeh, Pouria Fewzee, Ali Ghodsi, Mohamed S Kamel, and Fakhri Karray, *Multiview supervised dictionary learning in speech emotion recognition*, IEEE/ACM Transactions on Audio, Speech, and Language Processing **22** (2014), no. 6, 1056–1068.

[GFGK15] Mehrdad J Gangeh, Ahmed K Farahat, Ali Ghodsi, and Mohamed S Kamel, *Supervised dictionary learning and sparse representation-a review*, arXiv preprint arXiv:1502.05928 (2015).

[HKR16] Seung-Yeal Ha, Hwa Kil Kim, and Sang Woo Ryoo, *Emergence of phase-locked states for the kuramoto model in a large coupling regime*, Commun. Math. Sci **14** (2016), 1073–1091.

[JMB04] Ali Jadbabaie, Nader Motee, and Mauricio Barahona, *On the stability of the kuramoto model of coupled nonlinear oscillators*, American Control Conference, 2004. Proceedings of the 2004, vol. 5, IEEE, 2004, pp. 4296–4301.

[KB12] Johannes Klinglmayr and Christian Bettstetter, *Self-organizing synchronization with inhibitory-coupled oscillators: Convergence and robustness*, ACM Transactions on Autonomous and Adaptive Systems (TAAS) **7** (2012), no. 3, 30.

[KGC+18] Daniel S Kermany, Michael Goldbaum, Wenjia Cai, Carolina CS Valentim, Huiying Liang, Sally L Baxter, Alex McKeown, Ge Yang, Xiaokang Wu, Fangbing Yan, et al., *Identifying medical diagnoses and treatable diseases by image-based deep learning*, Cell **172** (2018), no. 5, 1122–1131.

[KPO+16] Donghyun Kim, Chanyoung Park, Jinoh Oh, Sungyoung Lee, and Hwanjo Yu, *Convolutional matrix factorization for document context-aware recommendation*, Proceedings of the 10th ACM conference on recommender systems, 2016, pp. 233–240.

[Kur75] Yoshiki Kuramoto, *Self-entrainment of a population of coupled non-linear oscillators*, International symposium on mathematical problems in theoretical physics, Springer, 1975, pp. 420–422.

[Kur03] _____, *Chemical oscillations, waves, and turbulence*, Courier Corporation, 2003.

[LK+20] Hanbaek Lyu, Yacoub Kureh, , Josh Vendrow, and Mason Porter, *Learning low-rank latent mesoscale structures in networks*, In preparation (2020).

[LLY22] Joowon Lee, Hanbaek Lyu, and Weixin Yao, *Supervised dictionary learning with auxiliary covariates*, arXiv preprint arXiv:2206.06774 (2022).

[LNB20] Hanbaek Lyu, Deanna Needell, and Laura Balzano, *Online matrix factorization for markovian data and applications to network dictionary learning*, Journal of Machine Learning Research 21 (To appear) (2020).

[LS99] Daniel D Lee and H Sebastian Seung, *Learning the parts of objects by non-negative matrix factorization*, Nature **401** (1999), no. 6755, 788.

[LS01] _____, *Algorithms for non-negative matrix factorization*, Advances in neural information processing systems, 2001, pp. 556–562.

[LSF+19] Johannes Leuschner, Maximilian Schmidt, Pascal Fernsel, Delf Lachmund, Tobias Boskamp, and Peter Maass, *Supervised non-negative matrix factorization methods for maldi imaging applications*, Bioinformatics **35** (2019), no. 11, 1940–1947.

[Lyu15] Hanbaek Lyu, *Synchronization of finite-state pulse-coupled oscillators*, Physica D: Nonlinear Phenomena **303** (2015), 28–38.

[Lyu16] ———, *Phase transition in firefly cellular automata on finite trees*, arXiv preprint arXiv:1610.00837 (2016).

[Mai13a] Julien Mairal, *Optimization with first-order surrogate functions*, International Conference on Machine Learning, 2013, pp. 783–791.

[Mai13b] ———, *Stochastic majorization-minimization algorithms for large-scale optimization*, Advances in Neural Information Processing Systems, 2013, pp. 2283–2291.

[MBP11] Julien Mairal, Francis Bach, and Jean Ponce, *Task-driven dictionary learning*, IEEE transactions on pattern analysis and machine intelligence **34** (2011), no. 4, 791–804.

[MBPS10] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro, *Online learning for matrix factorization and sparse coding*, Journal of Machine Learning Research **11** (2010), no. Jan, 19–60.

[McK] Brendan McKay, *Graphs*, http://users.cecs.anu.edu.au/~bdm/data/graphs.html.

[MPS+08] Julien Mairal, Jean Ponce, Guillermo Sapiro, Andrew Zisserman, and Francis Bach, *Supervised dictionary learning*, Advances in Neural Information Processing Systems **21** (2008), 1033–1040.

[MS90] Renato E Mirollo and Steven H Strogatz, *Synchronization of pulse-coupled biological oscillators*, SIAM Journal on Applied Mathematics **50** (1990), no. 6, 1645–1662.

[NL07] Sujit Nair and Naomi Ehrich Leonard, *Stable synchronization of rigid body networks*, Networks & Heterogeneous Media **2** (2007), no. 4, 597.

[PSC09] Vidyasagar Potdar, Atif Sharif, and Elizabeth Chang, *Wireless sensor networks: A survey*, Advanced Information Networking and Applications Workshops, 2009. WAINA'09. International Conference on, IEEE, 2009, pp. 636–641.

[RBK+20] Alexander Ritchie, Laura Balzano, Daniel Kessler, Chandra S Sripada, and Clayton Scott, *Supervised pca: A multiobjective approach*, arXiv preprint arXiv:2011.05309 (2020).

[SBK05] Bharath Sundararaman, Ugo Buy, and Ajay D Kshemkalyani, *Clock synchronization for wireless sensor networks: a survey*, Ad Hoc Networks **3** (2005), no. 3, 281–323.

[SG07] Mark Steyvers and Tom Griffiths, *Probabilistic topic models*, Handbook of latent semantic analysis **427** (2007), no. 7, 424–440.

[Str00] Steven H Strogatz, *From kuramoto to crawford: exploring the onset of synchronization in populations of coupled oscillators*, Physica D: Nonlinear Phenomena **143** (2000), no. 1-4, 1–20.

[WND13] Yongqiang Wang, Felipe Nunez, and Francis J Doyle, *Increasing sync rate of pulse-coupled oscillators via phase response function design: theory and application to wireless networks*, Control Systems Technology, IEEE Transactions on **21** (2013), no. 4, 1455–1462.

[YE17] Yael Yankelevsky and Michael Elad, *Structure-aware classification using supervised dictionary learning*, 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2017, pp. 4421–4425.

[ZHL+15] Shijie Zhao, Junwei Han, Jinglei Lv, Xi Jiang, Xintao Hu, Yu Zhao, Bao Ge, Lei Guo, and Tianming Liu, *Supervised dictionary learning for inferring concurrent brain networks*, IEEE transactions on medical imaging **34** (2015), no. 10, 2036–2045.

[ZL10] Qiang Zhang and Baoxin Li, *Discriminative k-svd for dictionary learning in face recognition*, 2010 IEEE computer society conference on computer vision and pattern recognition, IEEE, 2010, pp. 2691–2698.