Scalable Kernel Learning, Tensors in Community Identification, and Robust Adversary Detection in Deep Neural Networks

A DISSERTATION
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Fatemeh Sheikholeslami

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Prof. Georgios B. Giannakis, Advisor

August, 2019
Acknowledgments

Foremost, I would like to express my sincere gratitude to my advisor Prof. Georgios B. Giannakis for providing me with the opportunity to be a part of the SPiNCOM research group, as well as the ECE/CS graduate program of University of Minnesota. He has helped me in developing clear and scientific thought and expression, and without his guidance, the completion of this PhD Thesis would not have been possible.

I would like to extend my gratitude to Prof. Mostafa Kaveh, Prof. Zhili Zhang, and Prof. George Karypis, who agreed to serve as my thesis committee, and provided with much appreciated feedback that helped improve this work. Thanks go to a number of other professors in the departments of Electrical Engineering and Computer Science whose graduate level courses provided me with the necessary background to embark on this area of research.

Special thanks to my dear colleagues and friends, Dr. Dimitrios Berberidis, Dr. Antonio Marques, Alireza Sadeghi, Dr. Panagiotis Traganitis, Charilaos Kanatsoulis, and Donghoon Lee as well as all my other lab-mates in SPINCOM for their support and fruitful discussions and collaborations over the years.

Last but not least, I would like to express my deepest thanks to my parents and sister, for their unconditional love and support throughout my life.

Fateme Sheikholeslami, Minneapolis, Summer, 2019.
Dedication

This dissertation is dedicated to my family members for their unconditional love and support.
Abstract

The presence of ubiquitous sensors continuously recording massive amounts of information has lead to an unprecedented data collection, whose exploitation is expected to bring about scientific and social advancements in everyday lives. Along with the ever-increasing amount of data, incredible progress in the fields of Machine Learning, Pattern Recognition, and Optimization has also contributed to the growing expectations. Such progress however, has also brought to light certain limitations in state-of-the-art learning machines, manifesting the roadblocks in the research path ahead. For instance, in addition to practical considerations pertaining to non-stationary, noisy and unsupervised settings, various applications often run on limited memory and stringent computational resources, thus requiring efficient and light-weight algorithms to cope with extreme volumes. Furthermore, certain characteristics such as presence of outliers or adversaries as well as the complex nature of real-world interactions call for robust algorithms, whose performance will be resilient in the face of deviations from nominal settings.

The present thesis contributes to learning over unsupervised, complex, and adversarial data. Emphasis is laid on concocting online, scalable and robust algorithms, enabling streaming analytics of sequential measurements based on vector, matrix, and tensor-based views of supervised and unsupervised learning tasks. For online and scalable learning, a novel kernel-based feature extraction framework is put forth, in which limited memory and computational resources are accounted for via maintaining an affordable budget. Furthermore, complex interactions of real-world networks are analyzed from a community identification point-of-view, in which a novel tensor-based representation along with provable optimization techniques robustify state-of-the-art alternatives. Finally, the performance of deep convolutional neural network based image classifiers is investigated when adversaries disturbing input images are modeled as imperceptible yet carefully-crafted perturbations. To this end, a general class of high-performance Bayesian detectors of adversaries is developed. Extensive experimentation on synthetic as well as numerous real datasets demonstrates the effectiveness, interpretability and scalability of the proposed learning, identification, and detection algorithms. More importantly, the process of design and experimentation sheds light on the behavior of different methods and the peculiarities of real-world data, while at the same time it generates new ideas and directions to be explored.
Contents

Acknowledgments i

Dedication ii

Abstract iii

List of Tables viii

List of Figures x

1 Introduction 1

1.1 Scalable kernel-based feature extraction on a budget ... 2
   1.1.1 Prior art ................................................. 2
   1.1.2 Our contribution ........................................ 4

1.2 Robust identification of communities in complex networks ... 5
   1.2.1 Prior art .................................................. 5
   1.2.2 Our contributions ....................................... 6

1.3 Efficient Bayesian detection of adversarial attacks in DNNs ... 7
   1.3.1 Prior art .................................................. 8
   1.3.2 Our contributions ....................................... 9

1.4 Thesis outline ............................................... 9
1.5 Notational conventions ....................................... 10

2 Online Kernel-based Feature Extraction via Budgeted Subspace Tracking 11

2.1 Preliminaries and problem statement .......................... 11
# Minimum Uncertainty Based Detection of Adversaries in Deep Neural Networks

## 4 Minimum Uncertainty Based Detection of Adversaries in Deep Neural Networks

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Bayesian neural network preliminaries</td>
<td>60</td>
</tr>
<tr>
<td>4.1.1</td>
<td>Variational inference</td>
<td>61</td>
</tr>
<tr>
<td>4.1.2</td>
<td>Bayesian detection of DNN adversaries</td>
<td>62</td>
</tr>
<tr>
<td>4.2</td>
<td>Minimum Uncertainty-based Detection</td>
<td>63</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Uncertainty measures</td>
<td>64</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Simplification of the predictive variance</td>
<td>66</td>
</tr>
<tr>
<td>4.2.3</td>
<td>Layer-wise variance minimization</td>
<td>67</td>
</tr>
<tr>
<td>4.3</td>
<td>Solving layer-by-layer minimization</td>
<td>68</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Approximate variance minimization for small ( C )</td>
<td>69</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Approximate variance minimization for large ( C )</td>
<td>70</td>
</tr>
<tr>
<td>4.4</td>
<td>Practical issues</td>
<td>70</td>
</tr>
<tr>
<td>4.4.1</td>
<td>Efficient implementation via non-uniform dropout</td>
<td>71</td>
</tr>
<tr>
<td>4.4.2</td>
<td>Placement and adjustment of the sampling units</td>
<td>71</td>
</tr>
<tr>
<td>4.5</td>
<td>Numerical tests</td>
<td>75</td>
</tr>
<tr>
<td>4.5.1</td>
<td>CIFAR10 dataset</td>
<td>76</td>
</tr>
<tr>
<td>4.5.2</td>
<td>Cats-and-dogs dataset</td>
<td>82</td>
</tr>
</tbody>
</table>

## 5 Summary and Future Directions

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Thesis summary</td>
<td>85</td>
</tr>
<tr>
<td>5.2</td>
<td>Future research</td>
<td>86</td>
</tr>
<tr>
<td>5.2.1</td>
<td>Kernel-based representation of networks</td>
<td>86</td>
</tr>
<tr>
<td>5.2.2</td>
<td>Joint utilization of egonet-tensors and extra nodal features</td>
<td>87</td>
</tr>
<tr>
<td>5.2.3</td>
<td>Multi-dimensional egonet-tensors for intruder detection</td>
<td>88</td>
</tr>
<tr>
<td>5.2.4</td>
<td>Smart egonet-sampling and collapsing over large graphs</td>
<td>88</td>
</tr>
<tr>
<td>5.2.5</td>
<td>Ensemble of detection networks against adversarial inputs in CNNs</td>
<td>89</td>
</tr>
</tbody>
</table>

## References

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Appendix A. Proofs for Chapter 2</td>
<td>Proof of Proposition 2</td>
<td>105</td>
</tr>
</tbody>
</table>
## Appendix B. Proofs for Chapter 4

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>B.1 Proof of Proposition 1</td>
<td>107</td>
</tr>
<tr>
<td>B.2 Proof of Proposition 2</td>
<td>109</td>
</tr>
<tr>
<td>B.3 Selection of defense parameters</td>
<td>110</td>
</tr>
</tbody>
</table>
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Mean and variance of LS-fitting error of the extracted features with ((r, B) = (10, 15)) for different activities using different budget maintenance strategies.</td>
<td>39</td>
</tr>
<tr>
<td>2.2</td>
<td>Specifications of datasets.</td>
<td>41</td>
</tr>
<tr>
<td>3.1</td>
<td>Real-world networks.</td>
<td>57</td>
</tr>
<tr>
<td>3.2</td>
<td>Coverage and number of detected communities of different methods over real-world networks.</td>
<td>58</td>
</tr>
<tr>
<td>4.1</td>
<td>CIFAR10 and cats-and-dogs image-classification datasets.</td>
<td>76</td>
</tr>
<tr>
<td>4.2</td>
<td>ResNet20 architecture on CIFAR10 dataset</td>
<td>76</td>
</tr>
<tr>
<td>4.3</td>
<td>ResNet34 architecture on cats-and-dogs dataset</td>
<td>77</td>
</tr>
<tr>
<td>4.4</td>
<td>Attack parameters and test accuracy on clean, FGSM and BIM adversarial input in CIFAR10 dataset.</td>
<td>79</td>
</tr>
<tr>
<td>4.5</td>
<td>Attack parameters and test accuracy on BIM and C&amp;W adversarial input in CIFAR10.</td>
<td>79</td>
</tr>
<tr>
<td>4.6</td>
<td>AUC-ROC of different attack-detection sampling schemes on CIFAR10 test set against FGSM and MIM attacks. Higher values indicate better detection.</td>
<td>79</td>
</tr>
<tr>
<td>4.7</td>
<td>AUC-ROC of different attack-detection sampling schemes on CIFAR10 test set against FGSM and MIM attacks. Higher values indicate better detection.</td>
<td>81</td>
</tr>
<tr>
<td>4.8</td>
<td>AUC-ROC of different attack-detection sampling schemes on CIFAR10 test set against FGSM, C&amp;W, and combination attacks. Higher values indicate better detection.</td>
<td>81</td>
</tr>
<tr>
<td>4.9</td>
<td>AUC-ROC of different attack-detection sampling schemes on CIFAR10 test set against FGSM, C&amp;W, and combination attacks. Higher values indicate better detection.</td>
<td>81</td>
</tr>
</tbody>
</table>
4.10 Attack parameters and test accuracy on clean and adversarial input in cats-and-dogs dataset.

4.11 AUC-ROC of different attack-detection sampling schemes on cats-and-dogs dataset with against FGSM attack. Higher values indicate better detection.

4.12 AUC-ROC of different attack-detection sampling schemes on cats-and-dogs dataset with against MIM attack. Higher values indicate better detection.

4.13 AUC-ROC of different attack-detection sampling schemes on cats-and-dogs test set against BIM attack. Higher values indicate better detection.

4.14 AUC-ROC of different attack-detection sampling schemes on cats-and-dogs test set against C&W and combination attacks. Higher values indicate better detection.
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Toy example on how non-linear kernel-based feature mapping can transform data vectors for efficient linear learning in the lifted space.</td>
<td>3</td>
</tr>
<tr>
<td>1.2</td>
<td>Detection of overlapping communities in a brain network using fMRI measurements. All 90 brain regions were organized into 5 overlapping communities painted with different colors. Shared nodes are indicated by square symbols (red color), in which the large squares are shared by three communities, and the small squares are shared by two communities. Connections within the same community are painted with the color of the community. Connections between communities are painted with gray. For a detailed description, see [135].</td>
<td>6</td>
</tr>
<tr>
<td>1.3</td>
<td>Carefully crafted adversarial attacks on street signs can fool CNN classifiers [114].</td>
<td>8</td>
</tr>
<tr>
<td>2.1</td>
<td>LS-fit versus iteration index for the synthetic dataset (a), and convergence curve for the subspace iterates (b).</td>
<td>35</td>
</tr>
<tr>
<td>2.2</td>
<td>LS-fit of OKFE for different choices of polynomial and Gaussian kernels with different parameters.</td>
<td>36</td>
</tr>
<tr>
<td>2.3</td>
<td>LS-fit for different choices of $(r, B)$ using OK-FEB.</td>
<td>36</td>
</tr>
<tr>
<td>2.4</td>
<td>Visualization of the nonlinear synthetic manifolds.</td>
<td>37</td>
</tr>
<tr>
<td>2.5</td>
<td>LS-fitting error of dynamic dataset versus time.</td>
<td>37</td>
</tr>
<tr>
<td>2.6</td>
<td>Average kernel mismatch of dynamic data.</td>
<td>38</td>
</tr>
<tr>
<td>2.7</td>
<td>LS-fitting error of the PAMAP2 dataset versus time.</td>
<td>39</td>
</tr>
<tr>
<td>2.8</td>
<td>Average kernel mismatch of PAMAP2 dataset.</td>
<td>40</td>
</tr>
<tr>
<td>2.9</td>
<td>Online classification tests on (a) Adult, and regression tests on (b) CADATA, (c) Slice, and (d) Year datasets.</td>
<td>42</td>
</tr>
<tr>
<td>3.1</td>
<td>Construction of the three-way egonet-tensor.</td>
<td>45</td>
</tr>
<tr>
<td>Section</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>3.2</td>
<td>(a) A toy network with 5 non-overlapping communities; (b) corresponding egonet-tensor; and (c) its community-revealing factorization via PARAFAC decomposition.</td>
<td></td>
</tr>
<tr>
<td>3.3</td>
<td>The proposed DC-EgoTen community detection algorithm on a toy example.</td>
<td></td>
</tr>
<tr>
<td>3.4</td>
<td>Performance of different algorithms versus different community mixing values ( \mu ) for ( o_n = 600 ), and ( o_m = 3 ).</td>
<td></td>
</tr>
<tr>
<td>3.5</td>
<td>Performance of different algorithms versus different number of overlapping nodes ( o_n ) for ( \mu = 0.2 ), and ( o_m = 3 ).</td>
<td></td>
</tr>
<tr>
<td>3.6</td>
<td>Runtime of different algorithms on various datasets denoted on the x-axis as: (D1) Facebook, (D2) Enron, (D3) Epinion, (D4) Email, (D5) Slashdot, (D6) Notredame, (D7) Stanford, and (D8) Youtube.</td>
<td></td>
</tr>
<tr>
<td>3.7</td>
<td>Conductance-coverage curve for various datasets using different community detection algorithms.</td>
<td></td>
</tr>
<tr>
<td>4.1</td>
<td>Overview of Bayesian adversary detection schemes</td>
<td></td>
</tr>
<tr>
<td>4.2</td>
<td>Schematic of the proposed detection schemes</td>
<td></td>
</tr>
<tr>
<td>4.3</td>
<td>ROC-curve of different attack-detection sampling schemes on CIFAR10 dataset.</td>
<td></td>
</tr>
<tr>
<td>4.4</td>
<td>ROC-curve of different attack-detection sampling schemes on cats-and-dogs dataset.</td>
<td></td>
</tr>
<tr>
<td>B.1</td>
<td>Performance of different sampling mechanisms at various depths and parameters against combination attack on the CIFAR10 dataset.</td>
<td></td>
</tr>
<tr>
<td>B.2</td>
<td>Performance of different sampling mechanisms at various depths and parameters against combination attack on the cats-and-dogs dataset.</td>
<td></td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

The presence of ubiquitous sensors continuously collecting and recording massive amounts of information has lead to the present era of data deluge. Learning from these dynamics and huge volumes of data is expected to bring significant technological advances along with consequent improvements in quality of human life. With such big blessings however, come big challenges. In addition to practical considerations pertaining to noisy measurements, non-stationarity, and the presence of anomalies and outliers, various applications often run on unsupervised large-scale datasets. Dealing with such data, the goal of a learning algorithm is to find patterns in order to explain real-world phenomena while no training samples are provided. To this end, the sheer volume of data along with limited memory and computational capacities of current processing units require development of efficient and light-weight algorithms that can handle extreme volumes. Finally, certain characteristics of the data such as presence of outliers or adversaries as well as the complex nature of real-world interactions call for robust algorithms, whose performance will be resilient to deviations from nominal models.

This thesis contributes to learning from unsupervised, complex, and adversarial data. Emphasis is laid on concocting online, scalable, and robust algorithms to enable streaming analytics of sequential measurements based on vector, matrix, and tensor-based views of supervised and unsupervised learning tasks. The proposed research will draw from recent advances in optimization, machine learning, inference, and tensor analysis to markedly improve state-of-the-art performance.

Specifically, this thesis investigates three contemporary issues in online, scalable, and robust learning:
1.1 Scalable kernel-based feature extraction on a budget;

1.2 Robust identification of communities in complex networks; and,

1.3 Efficient Bayesian detection of adversarial attacks in DNNs.

Next, we outline state-of-the-art approaches in these areas, along with associated challenges and the contributions offered by the present thesis.

1.1 Scalable kernel-based feature extraction on a budget

Pattern recognition and machine learning target identification and training of mathematical models that can afford interpretability, as well as means of capturing real-world interactions. For instance, observations of protein-protein interactions help in understanding gene-regulatory networks enabling advanced medical treatments. Likewise, brain signals recorded by functional magnetic resonance imagining (fMRI) or the electroencephalogram (EEG) measurements can aid in understanding the dynamics behind brain networks. Other sources of data also include electronic transactions present in E-commerce, as well as data collected through the Internet and social media interactions. As linear models fail in modeling such a wide variety of signals, nonlinear models including those expressed using kernel-based expansions boost the learning and generalization capabilities in state-of-the-art methods; see Fig. 1.1.

When provided with sufficient training data, kernel methods can approximate arbitrary nonlinear functions with desired accuracy. Although “data deluge” sets the stage by providing “data-hungry” kernel methods with huge datasets, limited memory and computational constraints prevent such tools from fully exploiting their learning capabilities. In particular, given $N$ training $D \times 1$ vectors $\{x_\nu\}_{\nu=1}^N$, kernel regression or classification machines take $O(N^2 D)$ operations to form the $N \times N$ kernel matrix $K$, memory $O(N^2)$ to store it, and $O(N^3)$ computational complexity to find the sought predictor or classifier. For large datasets or online streams with increasing $N$, such computational and memory requirements will quickly exceed available resources, rendering a large class of nonlinear machines not scalable in practice.

1.1.1 Prior art

In this context, several efforts have been made in different fields of stochastic optimization, functional analysis, and numerical linear algebra to speed up kernel machines for “big data”
Figure 1.1: Toy example on how non-linear kernel-based feature mapping can transform data vectors for efficient linear learning in the lifted space.

Applications [63, 21, 106, 27, 134, 97, 76]. A common approach to scaling up kernel methods is to approximate the kernel matrix $K$ by a low-rank factorization; that is, $K \approx \hat{K} := Z^T Z$, where $Z \in \mathbb{R}^{r \times N}$ with $r (\ll N)$ is the reduced rank, through which storage and computational requirements go down to $O(Nr)$ and $O(Nr^2)$, respectively. Kernel (K)PCA provides a viable factorization for such low-rank approximation, at the cost of order $O(N^2r)$ computations [104]. Alternatively, a low-rank factorization can be effected by randomly selecting $r$ training vectors to approximate the kernel matrix [70]. Along these lines, Nystrom approximation [134], and its advanced renditions [27, 143, 66, 111, 127] are popular among this class of randomized factorizations. They trade off accuracy in approximating $K$ with $\hat{K}$, for reducing KPCA complexity from $O(N^2r)$ to $O(Nr)$. Their merits are well documented for nonlinear regression and classification tasks performed offline [19, 138, 5].

Rather than factorizing $K$, one can start from high-dimensional (lifted) feature vectors $\phi(x_\nu)$ whose inner product induces the kernel $\kappa(x_i, x_j) := \langle \phi(x_i), \phi(x_j) \rangle$ [97, 142, 74, 76, 108]. Approximating $\phi(x)$ by an $r \times 1$ vector $z$, any kernel can be approximated by a linear one as $\kappa(x_i, x_j) \approx z_i^T z_j$. Exploiting fast linear learning machines [33, 106], the kernel-based task then reduces to learning a linear function over $\{z_\nu\}_{\nu=1}^N$, which can be achieved in $O(Nr)$ operations. Such a computationally attractive trick is common to both kernel matrix factorization and lifted
feature approximation. Note however, that online Nystrom-type schemes are not available, while feature approximation algorithms are randomized, and thus they are not data driven.

In addition to batch kernel matrix and feature approximations, online kernel-based algorithms are of paramount importance. Instead of loading the entire datasets in memory, online methods iteratively pass over the set from an external memory [63, 106, 121, 14, 57, 105, 62]. This is also critical when the entire dataset is not available beforehand, but is acquired one datum at a time. For large data streams however, as the number of data increases with time, the support vectors (SVs) through which the function is estimated, namely the set $S$ in the approximation $f(x) \simeq \hat{f}(x) = \sum_{i \in S} \alpha_i K(x_i, x)$, also increases in size. Thus, the function evaluation delay as well as the required memory for storing the SV set eventually become unaffordable. Efforts have been devoted to reducing the number of SVs while maintaining performance on unseen data (a.k.a. generalization capability) [23]. By restricting the maximum number of SVs to a predefined budget $B$, the growth of algorithmic complexity is confined to an affordable limit, that is maintained throughout the online classification [129, 128, 22] or regression [123] tasks.

1.1.2 Our contribution

In Chapter 2 of this thesis, a generative model is built, according to which the high (possibly infinite)-dimensional features are approximated by their projection onto a low-rank subspace, thus providing a linear kernel function approximation. In contrast to [97, 27, 143, 74], where due to the nature of randomization the number of required features for providing an accurate kernel function approximation is often large, systematically learning the ambient nonlinear subspace yields an accurate approximation through a smaller number of extracted features. In order to keep the complexity and memory requirements affordable, budgeted versions of the proposed algorithms are devised in Section 2.4 in which the number of stored data vectors is confined to a predefined budget $B$. Budget maintenance is performed through a greedy approach, whose effectiveness is corroborated in simulated tests. This is the first work to address dynamic nonlinear (kernel-based) feature extraction under limited memory resources.

Analytical results provide performance bounds on kernel matrix approximation, and the resultant performance gap of budgeted kernel-based classification and regression approximations from their optimal unbudgeted counterparts. Furthermore, offline and online solvers for the proposed budgeted subspace learning are developed, and their convergence is analyzed. Finally, experiments on synthetic and real datasets demonstrate the efficiency of the proposed methods in
1.2 Robust identification of communities in complex networks

Real-world interactions among various entities, such as brain cells, users in social media platforms including Facebook and Twitter, and researchers in a scientific community, often exhibit interesting characteristics, the study of which can lead to a deeper understanding of the unknown dynamics driving the functionality and evolution of these networks. To this end, network analysis via mathematical graphs provides a rich toolbox to model such complex interactions, and unveil a great amount of information in diverse fields. Among such findings is the fact that many real-world networks demonstrate distinct characteristics, including power-law degree distribution, the small-world phenomena, and the presence of densely connected sub-graphs, also referred to as “communities” or “clusters” [37].

Focusing on the latter, strong connectivity of a subset of network entities, or nodes, along with their sparse interactions with the rest of the network is indicative of a “real-world association” among the participating nodes. The task of community detection targets the discovery of such communities, whose identification is of paramount importance in diverse fields ranging from gene-regulatory networks [28], to brain functionality [95], and social-media evolution [72, 89], to name a few; see Fig. 1.2.

1.2.1 Prior art

Past works on community detection include those based on generative and statistical models [1, 3, 136], modularity and related local-metric optimization [24, 13, 29], spectral clustering [125], and matrix factorization approaches [126, 16, 96, 144, 145, 95]; see also [37] and [35] for comprehensive overviews. However, most existing works pursue a bottom-up approach, where small collections of nodes with strong connectivity patterns (e.g., cliques) are selected as “seeds,” and larger communities are “grown” around them by merging other (clusters of) nodes [24, 132]. In contrast, another class of algorithms follows a top-down perspective, where a graph is progressively broken into smaller pieces, out of which communities eventually emerge [87, 47, 98].

Recent exploratory studies have revealed new challenges over contemporary networks,
6

Figure 1.2: Detection of overlapping communities in a brain network using fMRI measurements. All 90 brain regions were organized into 5 overlapping communities painted with different colors. Shared nodes are indicated by square symbols (red color), in which the large squares are shared by three communities, and the small squares are shared by two communities. Connections within the same community are painted with the color of the community. Connections between communities are painted with gray. For a detailed description, see [135].

addressing the presence of overlapping communities [133, 55, 131], multimodal interaction of nodes over multiview networks [91, 90], exploitation of nodal and edge-related side-information [137], as well as dynamic interactions within a network [4, 7]. In tackling these challenges, tensors as multi-modal structures offer increased representational capacity, which can readily translate to improved performance [91, 90, 4, 58, 9, 65, 110].

1.2.2 Our contributions

Upon recognizing that a network is in fact the union of its egonets, that is the subgraph induced by a given node and its immediate neighbors, a novel network representation using multi-way data structures is advocated in Chapter 3. The premise here is that the proposed sparse tensor-based representation exhibits richer structure compared to the adjacency matrix representation of its matrix-based counterpart, and thus enables a more robust approach to community identification. To leverage this structure, a constrained tensor approximation framework is introduced using the parallel factor (PARAFAC) decomposition. The arising constrained trilinear optimization is handled via alternating minimization, where intermediate subproblems are solved using the alternating direction method of multipliers (ADMM) to ensure convergence. The factors obtained via the proposed EgoTen method provide soft community memberships, which can further be exploited for crisp, and possibly-overlapping community assignments.

A desirable characteristic of the proposed algorithm is its ability to trade off flexibility
for increased redundancy and memory cost. Nevertheless, the resulting tensor is extremely sparse, and off-the-shelf tools for sparse tensor computations can be readily utilized; see e.g., [64, 92, 116]. Moreover, by relying on a successive application of EgoTen, a novel top-down community detection approach, termed “divide-and-concur (DC) EgoTen,” is developed to enable successful utilization in large-scale real-world networks with unknown number of communities.

The upshot of our novel framework is threefold: i) the performance of community detection in complex networks improves markedly thanks to the rich structure of tensors; ii) construction of the egonet-tensor via parallel implementation and exploitation of sparsity endow the algorithm with scalability; and, iii) the proposed top-down approach offers communities with the desired resolution. In fact, many of the previously developed algorithms are susceptible to the so-termed “resolution limit” [36], where identification of very large communities reveals little information about the underlying graph structure. Finally, efficiency and robustness of the proposed approach are empirically demonstrated using several numerical tests on large real-world networks.

### 1.3 Efficient Bayesian detection of adversarial attacks in DNNs

Besides the nonlinear function estimation approach investigated in Chapter 2 and with the recent commercialization of graphical processing units (GPUs), there has been an increasing interest in exploitation of highly nonlinear deep neural networks (DNNs) for various machine learning tasks. Their unprecedented learning capability offered through deep and diverse structures have enabled state-of-the-art performance in tasks such as object recognition and detection [112, 100], speech recognition and language translation [119], voice synthesis [31], and many more, where DNNs reach or even surpass human-level accuracy. Despite their performance however, recent studies have cast doubt on the reliability of DNNs, as highly-accurate networks are shown to be extremely vulnerable to carefully crafted inputs designed to ‘fool’ them [120, 85, 122].

Such fragility can easily lead to sabotage once adversarial entities target critical environments such as autonomous cars [32], automatic speech recognition [140], and face detection [45, 15, 107]; see Fig. 1.3. Particularly with convolutional neural networks (CNN) for image classification, their extreme brittleness is highlighted since small adversarial perturbations on the clean data, although often imperceptible to the human eye, can cause the trained CNNs to classify the adversarial examples incorrectly with high confidence. The design of powerful adversarial perturbations has been thoroughly investigated in environments with different levels
Figure 1.3: Carefully crafted adversarial attacks on street signs can fool CNN classifiers \[114\].

of complexity and knowledge about the target CNN (white, grey, and black-box attacks) \[85, 17, 18, 93, 141\]. This in turn necessitates design of robust and powerful attack detection mechanisms for reliable and safe utilization of neural networks \[94\].

1.3.1 Prior art

Defense methods against adversarial perturbations have been pursued in two broad directions, namely attack detection and recovery schemes. Methods in the former category aim at detection of adversarial images by classifying the input into clean or adversarial classes, by utilizing tools such as auto-encoders \[49\], detection sub-networks \[83, 77\], and dropout units \[34\]. On the other hand, recovery schemes aim at robustifying the classification accuracy by data pre-processing \[50, 44\], adversarial training \[84, 103, 113\], sparsification of the network \[52, 42\] and Lipschitz regularization \[139, 46\], among other schemes.

Furthermore, over-confidence of deep neural networks in classifying “out-of-distribution,” i.e., samples which lie in unexplored regions in the input domain, or even “misclassified” samples, has been unraveled in \[56, 51\]. This has motivated the need for certainty estimation as well as calibration of the networks for a more reliable classification. To this end, modern Bayesian neural networks target this issue by modeling a distribution over the model weights \[79\], and estimating certainty of the network output through predictive entropy, variance, or mutual information \[115, 61, 34\]. The well-known dropout regularization technique is an example of such approximate Bayesian networks, and it is now widely used in training as well as testing of neural networks \[40, 39\].
Moreover, recent utilization of dropout has shown promising performance in successful detection of adversarial attacks, where other defense mechanisms fail [17]. In particular, [34] utilizes randomness of dropout units during the test phase as a defense mechanism, and approximates the classification uncertainty by Monte Carlo (MC) estimation of the output variance. Subsequently, images with high classification uncertainty are declared as adversary. Recently, dropout defense has been generalized to non-uniform sampling [25], where entries of the hidden-layer tensors are randomly sampled, with probabilities proportional to the entry values. The heuristic sampling method in [25] is inspired by intuitive reasoning: activation units with large entries have more information and should be sampled more often. However, mathematical understanding and connections with the known Bayesian framework have not been discussed.

1.3.2 Our contributions

We wish to expand our understanding of uncertainty estimation in neural networks, and subsequently improve the detection of adversarial inputs. The premise here is that inherent distance of the adversarial perturbation from the natural-image manifold will cause the overall network uncertainty to exceed that of the clean image, and thus successful detection can be obtained. To this end, we first express the overall uncertainty of the network in terms of the uncertainty in the hidden layers, and formulate the task of adversary detection as uncertainty minimization by optimizing over sampling probabilities in the hidden layers.

We show that the proposed layer-wise uncertainty minimization can afford an exact solver with super-linear convergence rate, as well as low-complexity approximate solvers. Furthermore, connections with uniform dropout [34] as well as stochastic approximate pruning (SAP) in [25] are drawn, and efficient implementation of the proposed method is provided by interpreting it as non-uniform dropout. The proposed approach has a desirable modular nature in its simplicity yet effectiveness: multiple sampling units can be placed at various depths in the network during the testing phase, thus robustifying the detection at no training overhead. Extensive tests on CIFAR10 and high-quality cats-and-dogs images in the presence of various attack schemes corroborate the importance of defense unit placement as well as tuning for improved detection.
1.4 Thesis outline

Chapter 2 of the present thesis deals with a low-complexity and scalable kernel-based nonlinear feature extraction algorithm. A generative model for approximating high (possibly infinite-)dimensional features by the proposed projection onto the low-rank subspace is developed. Offline and online solvers for the proposed subspace-tracking task over data streams along with their analytical convergence bounds are provided, and budgeted versions of the proposed algorithm are devised. Finally, numerical simulations on synthetic and real tests on the HAPMAP dataset corroborate the effectiveness of the proposed method.

Chapter 3 develops efficient and robust identification algorithms for overlapping communities in large-scale networks. EgoTen, our proposed tensor-based network representation along with the constrained trilinear optimization tuned towards overlapping community detection is introduced, and DC-EgoTen as its extension for large-scale networks is put forth. Discussions on performance metrics along with extensive numerical tests conclude this chapter.

Chapter 4 studies Bayesian detection of adversarial inputs in deep neural networks, where the focus is placed on the task of image classification in convolutional neural networks. An overview of Bayesian inference and detection in neural networks is provided, and a novel class of Bayesian detectors is introduced next. Exact and approximate solvers along with their efficient implementation are also provided. Effectiveness of the careful placement as well as tuning of the defense units for successful attack detection are highlighted via extensive numerical tests against various attack schemes in state-of-the-art networks.

Finally, concluding remarks are given in Chapter 5 along with future research directions.

1.5 Notational conventions

The following notation is used throughout the remainder of this thesis. Lower- (upper-) case boldface fonts denote vectors (matrices), while underlined uppercase bold letters stand for tensors. Calligraphic letters are reserved for sets, e.g., $\mathcal{S}$, and $^T$ stands for transposition. Symbols $\circ$ and $\otimes$ are reserved for outer- and Kronecker-product, respectively, while $\text{Tr}\{X\}$ denotes the trace of matrix $X$. Operator $\|\cdot\|_q$ denotes the $\ell_q$-norm on vector and matrix operands. Finally, $\succeq$ represents positive semi-definiteness of matrices, while the ordered eigenvalues of matrix $X \in \mathbb{R}^{n \times n}$ are given as $\lambda_1(X) \geq \lambda_2(X) \geq \cdots \geq \lambda_n(X)$. 
Chapter 2

Online Kernel-based Feature Extraction via Budgeted Subspace Tracking

2.1 Preliminaries and problem statement

Consider \( N \) real data vectors \( \{x_\nu\}_{\nu=1}^N \) of size \( D \times 1 \). As large values of \( D \) and \( N \) hinder storage and processing of such datasets, extracting informative features from the data (a.k.a. dimensionality reduction) results in huge savings on memory and computational requirements. This fundamentally builds on the premise that the informative part of the data is of low dimension \( r < D \), and thus the data \( \{x_\nu\}_{\nu=1}^N \) are well represented by the generative model

\[
x_\nu = Lq_\nu + v_\nu, \quad \nu = 1, \ldots, N
\]  

(2.1)

where the tall \( D \times r \) matrix \( L \) has rank \( r < D \); vector \( q_\nu \) is the \( r \times 1 \) projection of \( x_\nu \) onto the column space of \( L \); and \( v_\nu \) denotes zero-mean additive noise.

Pursuit of the subspace \( L \) and the low-dimensional features \( \{q_\nu\}_{\nu=1}^N \) is possible using a blind least-squares (LS) criterion regularized by a rank-promoting term using e.g., the nuclear norm of \( \hat{X} = LQ_N \), where \( Q_N := [q_1, \ldots, q_N] \). Albeit convex, nuclear-norm regularization is not attractive for sequential learning.

To facilitate reducing the computational complexity, it is henceforth assumed that an upper
bound on the rank of matrix $\hat{X}$ is given $\rho \geq \text{rank}(\hat{X})$. Thus, building on the work of [80] by selecting $r \geq \rho$, and to arrive at a scalable subspace tracker, here we surrogate the nuclear norm with the summation of the Frobenious-norms of $L$ and $Q_N$, which yields (cf. Prop. 1 in [80] for proof on equivalence)

$$\min_{L, \{q_{\nu}\}_{\nu=1}^N} \frac{1}{2N} \sum_{\nu=1}^n \|x_{\nu} - Lq_{\nu}\|_2^2 + \frac{\lambda}{2N} \left( \|L\|_F^2 + \|Q_N\|_F^2 \right)$$

where $\lambda$ controls the tradeoff between LS fit and rank regularization [81]. Principal component analysis (PCA) - the “workhorse” of dimensionality reduction- solves (2.2) when the rank regularization is replaced with orthonormality constraints on $L$. Undoubtedly, the accuracy of any linear dimensionality reduction method is dictated by how well the model (2.1) fits a given dataset, which is related to how well the corresponding data covariance matrix can be approximated by a low-rank matrix [53, p. 534].

In practice however, low-rank linear models often fail to accurately capture the datasets. A means to deal with nonlinearities in pattern recognition tasks, is to first map vectors $\{x_{\nu}\}_{\nu=1}^N$ to a higher $\bar{D}$-dimensional space using a function $\phi : \mathbb{R}^D \rightarrow \mathbb{R}^{\bar{D}}$ (possibly with $\bar{D} = \infty$), and subsequently seek a linear function over the lifted data $\phi(x)$. This map induces a so-called kernel function $\kappa(x_i, x_j) = \phi^\top(x_i)\phi(x_j)$. Selecting the kernel to have a closed-form expression circumvents the need to explicitly know $\{\phi(x_{\nu})\}_{\nu=1}^N$ - what is referred to as the “kernel trick.” Similarly, the norm corresponding to the reproducing kernel Hilbert space (RKHS) is defined as $\|\phi(x)\|_K^2 := \langle \phi(x), \phi(x) \rangle = \kappa(x, x)$. Upon defining the $\bar{D} \times N$ matrix $\Phi_N := [\phi(x_1), ..., \phi(x_N)]$, the $N \times N$ kernel matrix related to the covariance of the lifted data is formed with $(i, j)$ entry $\kappa(x_i, x_j)$ as $K(x_{1:N}, x_{1:N}) = \Phi_N^\top\Phi_N$, where $x_{1:N} := \text{vec}[x_1, x_2, ..., x_N]$. Its computation and storage incurs complexity $O(N^2D)$ and $O(N^2)$ respectively, which is often not affordable when $N \gg$ and/or $D \gg$.

Fortunately, $K$ for large data sets in practice has approximately low rank. This fact is exploited in e.g., [142, 27] and [134] to approximate $K$ via a low-rank factorization, hence reducing the evaluation and memory requirements of offline kernel-based learning tasks from $O(N^2D)$ down to $O(Nr)$. Here, we further build on this observation to deduce that the low-rank property of $K = \Phi_N^\top\Phi_N$ implies that $\Phi_N$ can also be approximated by a low-rank matrix,

\[1\] In practice, the rank is controlled by tuning regularization parameter, as it can be made small enough for sufficiently large $\lambda$.\]
thus motivating our pursuit of **online low-rank factorization** of $\Phi_N$. To this end, instead of projecting $\{x_\nu\}$s onto the columns of $L$ as in (2.2), we will project $\{\phi(x_\nu)\}$s on $\bar{L} \in \mathbb{R}^{D \times r}$, whose columns span what we refer to as “virtual” column subspace since $D$ can be infinite. Specifically, we consider [cf. (2.2)]

$$
\min_{L, (q_\nu)_{\nu=1}^N} \frac{1}{2N} \sum_{\nu=1}^N \|\phi(x_\nu) - \bar{L}q_\nu\|_H^2 + \frac{\lambda}{2N} \left( \|\bar{L}\|_{HS}^2 + \|Q_N\|_F^2 \right)
$$

(2.3)

where the $\ell_2$-norm has been substituted by the $H$-norm in the $\bar{D}$-dimensional Hilbert space. Similarly, let the Hilbert–Schmidt operator be defined as $\|\bar{L}\|_{HS} = \sqrt{\text{Tr}(\bar{L}^\top \bar{L})} := \sqrt{\sum_{c=1}^r \|\bar{l}_c\|_H^2}$ with $\bar{l}_c$ denoting the $c$-th column of $\bar{L}$. Note that for Euclidean spaces, the Hilbert-Schmidt norm reduces to the Frobenious norm.

Observe also that similar to the linear model in (2.2), upon removing the regularization terms and adding the orthonormality constraints on the columns of $\bar{L}$, (2.3) reduces to that of KPCA (without centering) in primal domain [104, p. 429]. The present formulation in (2.3) however, enables us to develop sequential learning algorithms, which will later be enhanced with a tracking capability for dynamic datasets.

For a fixed $Q_N$, the criterion in (2.3) is minimized by

$$
\bar{L}_N = \Phi_N Q_N^\top \left( Q_N Q_N^\top + \lambda I \right)^{-1} := \Phi_N A
$$

(2.4)

where the $N \times r$ factor $A$ can be viewed as “morphing” the columns of $\Phi_N$ to offer a flexible basis for the lifted data. Substituting (2.4) back into (2.3) and exploiting the kernel trick, we arrive at

$$
\min_{A, (q_\nu)_{\nu=1}^N} \frac{1}{2N} \sum_{\nu=1}^N \left( \kappa(x_\nu, x_\nu) - 2k^\top(x_{1:N}, x_\nu)Aq_\nu + q_\nu^\top A^\top K(x_{1:N}, x_{1:N})Aq_\nu \right) + \frac{\lambda}{2N} \left( \text{tr} \{ A^\top K(x_{1:N}, x_{1:N})A \} + \sum_{\nu=1}^N \|q_\nu\|_2^2 \right)
$$

(2.5)

where the $N \times 1$ vector $k(x_{1:N}, x_n)$ in (2.5) is the $n$-th column of $K(x_{1:N}, x_{1:N})$, and since $A$ has size $N \times r$, the minimization in (2.5) does not depend on $D$.

Our goal is to develop and analyze batch as well as online solvers for (2.5). By pre-specifying an affordable complexity for the online solver, we aim at a low-complexity algorithm where
subspace learning and feature extraction can be performed on-the-fly for streaming applications. Furthermore, we will introduce a novel approach to extracting features on which the kernel-based learning tasks of complexity $O(N^3)$ can be well approximated by linear counterparts of complexity $O(rN)$, hence realizing great savings in memory and computation while maintaining performance. A remark is now in order.

**Remark 1.** The subspace $L_N$ in (2.4) can be thought as a dictionary whose atoms are morphed via factor $A$. Sparse representation over kernel-based dictionaries have been considered [102, 48, 124, 82]. Different from these approaches however, the novelty here is on developing algorithms that can process streaming datasets, possibly with dynamic underlying generative models. Thus, our goal is to efficiently learn and track a dictionary that adequately captures streaming data vectors, and can afford a low-rank approximation of the underlying high-dimensional map.

### 2.2 Offline kernel-based feature extraction

Given a dataset $\{x_\nu\}_{\nu=1}^N$ and leveraging the bi-convexity of the minimization in (2.5), we introduce in this section a batch solver, where two blocks of variables ($A$ and $\{q_\nu\}_{\nu=1}^N$) are updated alternately. The following two updates are carried out iteratively until convergence.

**Update 1.** With $A[k]$ given from iteration $k$, the projection vectors $\{q_\nu\}_{\nu=1}^N$ in iteration $k + 1$ are updated as

$$q_\nu[k + 1] = \arg \min_q \ell(x_\nu; A[k], q; x_1:N) + \frac{\lambda}{2}\|q\|_2^2$$

(2.6a)

where the fitting cost $\ell(.)$ is given by [cf. (2.3)-(2.5)]

$$\ell(x_\nu; A[k], q; x_1:N) := \frac{1}{2}\|\phi(x_\nu) - \Phi_N A[k]q\|_H^2$$

(2.6b)

$$= \kappa(x_\nu, x_\nu) - 2k^T(x_1:N, x_\nu)A[k]q$$


The minimizer of (2.6a) yields the features as regularized projection coefficients of the lifted data vectors onto the virtual subspace $\tilde{L}_N[k] = \Phi_N A[k]$, and is given in closed form by

$$q_\nu[k + 1] = (A^T[k]K(x_1:N, x_1:N)A[k] + \lambda I_r)^{-1}$$
Algorithm 1 BKFE: Batch Kernel-based Feature Extraction

Input \(\{x_\nu\}_{\nu=1}^N, \lambda\)

Initialize \(A[1]\) at random

For \(k = 1, \ldots\) do

\[
S[k + 1] = (A^\top[k] K(x_{1:N}, x_{1:N}) A[k] + \lambda I_r)^{-1} A^\top[k]
\]

\[
Q[k + 1] = S[k + 1] K(x_{1:N}, x_{1:N})
\]

\[
A[k + 1] = Q_N^\top[k + 1] \left( Q_N[k + 1] Q_N^\top[k + 1] + \lambda I_r \right)^{-1}
\]

Repeat Until Convergence

Return \(A[k], \{q_\nu[k]\}_{\nu=1}^N\)

\[
\times A^\top[k] k(x_{1:N}, x_\nu), \quad \nu = 1, \ldots, N. \quad (2.7)
\]

**Update 2.** With \(\{q_\nu[k + 1]\}_{\nu=1}^N\) fixed and after dropping irrelevant terms, the subspace factor is obtained as [cf. (2.5)]

\[
A[k + 1] = \arg \min_A \frac{1}{N} \sum_{\nu=1}^N \ell(x_\nu; A, q_\nu[k + 1]; x_{1:N}) + \frac{\lambda}{2N} \text{tr}\{A^\top K(x_{1:N}, x_{1:N}) A\}.
\]

\[(2.8)\]

Since \(K\) is positive definite in practice, (2.8) involves a strictly convex minimization. Equating the gradient to zero, yields the wanted subspace factor in closed form

\[
A[k + 1] = Q_N^\top[k + 1] \left( Q_N[k + 1] Q_N^\top[k + 1] + \lambda I_r \right)^{-1}.
\]

\[(2.9)\]

Algorithm 1 provides the pseudocode for the update rules (2.7) and (2.9) of the batch solver, and the following proposition gives a guarantee on the convergence of the proposed solver to a local stationary point.

**Proposition 1.** For positive definite kernels and \(\lambda > 0\), the sequence \(\{A[k], Q_N[k]\}\) generated by Algorithm 1 converges to a stationary point of the minimization in (2.5).

**Proof:** Since the minimizations in (2.6a) and (2.8) are strictly convex with unique solutions, the result follows readily from [11, p. 272].

Since matrix inversions in (2.7) and (2.9) cost \(O(r^3)\), and \(Q_N\) and \(A\) have size \(r \times N\) and
$N \times r$, respectively, the per iteration cost is $O(N^2r + Nr^2 + r^3)$. Although the number of iterations needed in practice for Algorithm 1 to converge is effectively small, this per iteration complexity can be unaffordable for large datasets. In addition, datasets are not always available offline, or due to their massive volume, can not be uploaded into memory at once. To cope with these issues, an online solver for (2.5) is developed next, where the updates are carried out by iteratively passing over the dataset one datum at a time.

### 2.3 Online kernel-based feature extraction

This section deals with low-cost, on-the-fly updates of the ‘virtual’ subspace $\tilde{L}$, or equivalently its factor $A$ as well as the features $\{q_\nu\}$ that are desirable to keep up with streaming data. For such online updates, stochastic gradient descent (SGD) has well-documented merits, especially for parametric settings. However, upon processing $n$ data vectors, $A$ has size $n \times r$, which obviously grows with $n$. Hence, as the size of $A$ increases with the number of data, the task of interest is a nonparametric one. Unfortunately, performance of SGD on nonparametric learning such as the one at hand is an uncharted territory. Nevertheless, SGD can still be performed on the initial formulation (2.3), where solving for the virtual $\tilde{L}$ constitutes a parametric task, not dependent on $n$.

Starting with an update for $\tilde{L}$, an update for $A$ will be derived first, as an alternative to those in [106, 21], and [128]. Next, an SGD iteration for $A$ will be developed in subsection 2.3.2, while in subsection 2.3.3 a connection between the two update rules will be drawn, suggesting how SGD can be broadened to learning nonparametric models as well.

#### 2.3.1 SGD on “parametric” subspace tracking

Suppose that $x_n$ is acquired at time $n$, posing the overall joint subspace tracking and feature extraction problem as [cf. (2.3)]

$$
\min_{\tilde{L}, \{q_\nu\}} \frac{1}{2n} \sum_{\nu=1}^{n} \| \phi(x_\nu) - \tilde{L}q_\nu \|^2_H + \frac{\lambda}{2n} \left( \| \tilde{L} \|^2_H + \| Q_n \|^2_F \right). \tag{2.10}
$$

Using an alternating minimization approach, we update features and the subspace per data vector as follows.
Update 1. Fixing the subspace estimate at its recent value $\bar{L}_{n-1} := \Phi_{n-1} A_{n-1}$ from time $n-1$, the projection vector of the new data vector $x_n$ is found as [cf. (2.6a)]

$$q[n] = \arg \min_q \ell(x_n; A[n-1], q; x_{1:n-1}) + \frac{\lambda}{2} \|q\|^2$$  \hspace{1cm} (2.11a)

which through the kernel trick readily yields

$$q[n] = (A^\top [n-1] K(x_{1:n-1}, x_{1:n-1}) A[n-1] + \lambda I_r)^{-1} A^\top [n-1] k(x_{1:n-1}, x_n) . \hspace{1cm} (2.11b)$$

Although (2.11b) can be done for all the previous features $\{q[\nu]\}_{\nu=1}^{n-1}$ as well, it is skipped in practice to prevent exploding complexity. In the proposed algorithm, feature extraction is performed only for the most recent data vector $x_n$.

Update 2. Having obtained $q[n]$, the subspace update is given by solving

$$\min_{\bar{L}} \frac{1}{n} \sum_{\nu=1}^{n} \bar{L}(x_{\nu}; \bar{L}, q[\nu])$$ \hspace{1cm} (2.12)

where $\bar{L}(x_{\nu}; \bar{L}, q[\nu]) := \frac{1}{2} \|\phi(x_{\nu}) - \bar{L} q[\nu]\|^2_H + \frac{\lambda}{2n} \|\bar{L}\|^2_{HS}$. Solving (2.12) as time evolves, becomes increasingly complex, and eventually unaffordable. If data $\{x_{\nu}\}_{\nu=1}^{n}$ satisfy the law of large numbers, then (2.12) approximates $\min_{\bar{L}} \mathbb{E}[\bar{L}(x_{\nu}; \bar{L}, q[\nu])]$, where expectation is with respect to the unknown probability distribution of the data. To reduce complexity of the minimization, one typically resorts to stochastic approximation solvers, where by dropping the expectation (or the sample averaging operator), the ‘virtual’ subspace update is

$$\bar{L}[n] = \bar{L}[n-1] - \mu_{n,L} G_n$$ \hspace{1cm} (2.13)

with $\mu_{n,L}$ denoting a preselected stepsize, and $G_n$ the gradient of the $n$-th summand in (2.12) given by

$$G_n := \nabla_{\bar{L}} \bar{L}(x_{\nu}; \bar{L}[n-1], q[n])$$

$$= -\left( \phi(x_{\nu}) - \bar{L}[n-1] q[n] \right) q^\top [n] + \frac{\lambda}{n} \bar{L}[n-1]$$

$$= \Phi_n \begin{bmatrix} A[n-1] q[n] q^\top [n] \\ -q^\top [n] \end{bmatrix} + \frac{\lambda}{n} \Phi_n \begin{bmatrix} A[n-1] \\ 0_{1 \times r} \end{bmatrix} . \hspace{1cm} (2.14)$$
Algorithm 2 Online kernel-based feature extraction with parametric update rule

**Input** \( \{x_\nu\}_{\nu=1}^n, \lambda \)

**Initialize** \( A[1] = 1_{1 \times r}, K(x_1, x_1) = \kappa(x_1, x_1) \)

For \( n = 2, \ldots \) do

\[
q[n] = \left( A^\top [n-1] K(x_{1:n-1}, x_{1:n-1}) A[n-1] + \lambda I_r \right)^{-1} A^\top [n-1] K(x_{1:n-1}, x_n) \]

\[
K(x_{1:n}, x_{1:n}) = \begin{bmatrix}
K(x_{1:n-1}, x_{1:n-1}) & k(x_{1:n-1}, x_n) \\
K^\top (x_{1:n-1}, x_n) & \kappa(x_n, x_n)
\end{bmatrix}
\]

\[
A[n] = \begin{bmatrix}
A[n-1] - \mu_{n,L} A[n-1] \left( q[n] q^\top [n] + \frac{\lambda}{n} I_r \right)
\end{bmatrix}
\]

Return \( A[n], \{q[\nu]\}_{\nu=2}^n \)

Because \( \bar{L}[n] \) has size \( \bar{D} \times r \) regardless of \( n \), iteration (2.13) is termed “parametric” Using (2.4) to rewrite \( \bar{L}[n] = \Phi_n A[n] \), and substituting into (2.13), yields

\[
\Phi_n A[n] = \Phi_n \begin{bmatrix}
A[n-1] \\
0_{1 \times r}
\end{bmatrix} - \mu_{n,L} \Phi_n \begin{bmatrix}
A[n-1] \left( q[n] q^\top [n] + \frac{\lambda}{n} I_r \right)
\end{bmatrix}
\]

which suggests the following update rule for factor \( A \)

\[
A[n] = \begin{bmatrix}
A[n-1] - \mu_{n,L} A[n-1] \left( q[n] q^\top [n] + \frac{\lambda}{n} I_r \right)
\end{bmatrix}
\]

Even though (2.16) is not the only iteration satisfying (2.15), it offers an efficient update of the factor \( A \). The update steps for the proposed parametric tracker are summarized as Algorithm 2. Note that the multiplication and inversion in (2.9) are avoided. However, per data vector processed, the kernel matrix is expanded by one row and one column, while the subspace factor \( A \) grows accordingly by one row.

### 2.3.2 SGD for “nonparametric” subspace tracking

In this subsection, the feature extraction rule in (2.11b) is retained, while the update rule (2.16) is replaced by directly acquiring the SGD direction along the gradient of the instantaneous objective.
term with respect to \( A \). Since, in contrast to the fixed-size matrix \( \bar{L} \), the number of parameters in \( A \) grows with \( n \), we refer to the solver developed in this subsection as a nonparametric subspace tracker. Furthermore, the connection between the two solvers is drawn in subsection 2.3.3 and convergence of the proposed algorithm is analyzed in subsection 2.3.4.

At time instance \( n \), subproblem (2.12) can be expanded using the kernel trick as

\[
\min_{A \in \mathbb{R}^{n \times r}} \frac{1}{n} \sum_{\nu=1}^{n} \mathcal{L}(x_\nu; A, q[\nu]; x_{1:n})
\]  

where \( \mathcal{L}(x_\nu; A, q[\nu]; x_{1:n}) := \ell(x_\nu; A, q[\nu]; x_{1:n}) + \frac{\lambda}{2n} \text{tr}\{A^\top K(x_{1:n}, x_{1:n})A\} \)  

with \( \ell(\cdot) \) given by (2.6b). Stochastic approximation solvers of (2.17) suggest the update

\[
A[n] = \left[ A[n-1] \ 0_{r \times 1}^\top \right] - \mu_{n,A} G_n
\]  

where \( \mu_{n,A} \) denotes the user-selected step size, and \( G_n \) denotes the gradient of the \( n \)-th summand in (2.17) with respect to \( A \) that is given by

\[
G_n := \nabla_A \mathcal{L}(x_n; [A^\top [n-1], 0_{r \times 1}]^\top, q[n]; x_{1:n})
\]

\[
= K(x_{1:n}, x_{1:n}) \left[ A[n-1] \ 0_{r \times 1}^\top \right] q[n]^\top + \frac{\lambda}{n} K(x_{1:n}, x_{1:n}) \left[ A[n-1] \ 0_{r \times 1}^\top \right].
\]  

Substituting (2.19b) into (2.19a) yields the desired update of \( A \) which together with (2.11b) constitute our nonparametric solver, tabulated under Algorithm 3.

2.3.3 Parametric vis-a-vis nonparametric SGD updates

Considering that \( \bar{L}[n] = \Phi_n A[n] \) holds for all \( n \), it is apparent from (2.19b) and (2.14) that \( G_n = \Phi_n^\top G_n \). The latter implies that the update rule in (2.19a) amounts to performing SGD on...
Algorithm 3 Online kernel-based feature extraction with nonparametric update rule

Input \( \{x_\nu\}_{\nu=1}^n, \lambda \)

Initialize \( A[1] = 1_{1 \times r}, K(x_1, x_1) = \kappa(x_1, x_1) \)

For \( n = 2, \ldots \) do

\[
q[n] = (A^T[n - 1]K(x_{1:n-1}, x_{1:n-1})A[n - 1] + \lambda I_r)^{-1} A^T[n - 1]k(x_{1:n-1}, x_n)
\]

\[
K(x_{1:n}, x_{1:n}) = \begin{bmatrix}
K(x_{1:n-1}, x_{1:n-1}) & k(x_{1:n-1}, x_n) \\
k^T(x_{1:n-1}, x_n) & \kappa(x_n, x_n)
\end{bmatrix}
\]

\[
G_n = K(x_{1:n}, x_{1:n}) [A[n - 1]0_{r \times 1}] q[n] q^T[n] - k(x_{1:n}, x_n)q^T[n]
+ \frac{\lambda}{n} K(x_{1:n}, x_{1:n}) \begin{bmatrix} A[n - 1] \\ 0 \end{bmatrix}
\]

\[
A[n] = \begin{bmatrix} A[n - 1] \\ 0 \end{bmatrix} - \mu_{n,A} G_n
\]

Return \( A[n], \{q[\nu]\}_{\nu=2}^n \)

\[\bar{L} \text{ with a matrix stepsize } D_n = \Phi_n \Phi_n^T, \text{ that is,} \]
\[\bar{L}[n] = \bar{L}[n - 1] - \mu_{n,A} D_n \bar{G}_n. \quad (2.20)\]

It is important to check whether this \( D_n \) constitutes a valid descent direction, which is guaranteed since
\[G_n^T D_n \bar{G}_n = H_n^T K^T(x_{1:n}, x_{1:n})K(x_{1:n}, x_{1:n})H_n \succeq 0 \quad (2.21)\]

where
\[H_n := \begin{bmatrix}
A[n - 1](q_n q_n^T + \frac{\lambda}{n} I_r) \\ -q_n^T
\end{bmatrix}.\]

For positive-definite e.g., Gaussian kernel matrices, we have \( G_n^T D_n \bar{G}_n > 0 \), which guarantees that \( -D_n \bar{G}_n \) is a descent direction \[11\] p. 35]. Leveraging this link, Algorithm \[3\] will be shown next to enjoy the same convergence guarantee as that of Algorithm \[2\].

Remark 2. Although the SGD solver in Algorithm \[3\] can be viewed as a special case of Algorithm \[2\], developing the parametric SGD solver in Algorithm \[2\] will allow us to analyze convergence of the two algorithms in the ensuing subsections.
2.3.4 Convergence analysis

The cost in (2.10) can be written as

\[ F_n(\bar{L}) := \frac{1}{n} \sum_{\nu=1}^{n} \min_{q} f_{\nu}(x_{\nu}; \bar{L}, q) \] (2.22)

with \( f_{\nu}(x_{\nu}, \bar{L}, q) := \bar{L}(x_{\nu}; \bar{L}, q) + (\lambda/2)\|q\|_2^2 \), and \( \bar{L} \) as in (2.12). Thus, the minimization in (2.10) is equivalent to \( \min_{\bar{L}} F_n(\bar{L}) \). To ensure convergence of the proposed algorithms, the following assumptions are adopted.

(A1) \{x_{\nu}\}_{\nu=1}^{n} independent identically distributed; and

(A2) The sequence \( \{\|\bar{L}[\nu]\|_{HS}\}_{\nu=1}^{\infty} \) is bounded.

Data independence across time is standard when studying the performance of online algorithms [81], while boundedness of the iterates \( \{\|\bar{L}[\nu]\|_{HS}\}_{\nu=1}^{\infty} \), corroborated by simulations, is a technical condition that simplifies the analysis, and in the present setting is provided due to the Frobenious-norm regularization. In fact, rewriting subspace update in Alg. 2 yields

\[ \bar{L}[n] = \bar{L}[n-1](I - \mu_{n,L}(q[n]q^T[n] + \lambda/nI_r)) + \mu_{n,L}\phi_nq^T, \]

which consists of: i) contraction of the most recent subspace iterate; and, ii) an additive term. Thus, with proper selection of the diminishing step size \( \mu_{n,L} \), A2 is likely to hold. The following proposition provides convergence guarantee for the proposed algorithm.

**Proposition 2.** Under (A1)-(A2), if \( \mu_{n,L} = 1/\bar{\gamma}_n \) with \( \bar{\gamma}_n := \sum_{\nu=1}^{n} \gamma_{\nu} \) and \( \gamma_{\nu} \geq \|\nabla^2 \bar{L}(x_{\nu}; \bar{L}, q_{\nu})\|_{H} \) \( \forall n \), then the subspace iterates in (2.13) satisfy \( \lim_{n \to \infty} \nabla F_n(\bar{L}[n]) = 0 \) almost surely; that is, \( Pr\{\lim_{n \to \infty} \nabla F_n(\bar{L}[n]) = 0\} = 1 \), thus the sequence \( \{\bar{L}[\nu]\}_{\nu=1}^{\infty} \) falls into the stationary point of (2.10).

**Proof:** Proof is inspired by [78], and a sketch of the required modifications can be found in the Appendix.

So far, we have asserted convergence of the SGD-based algorithm for the “virtual” \( \bar{L} \) provided by Algorithm 2. A related convergence result for Algorithm 3 is guaranteed by the following argument.

**Proposition 3.** Under (A1)-(A2) and for positive definite radial kernels, if \( \mu_{n,A} = 1/\bar{\xi}_n \) with \( \bar{\xi}_n := \sum_{\nu=1}^{n} \xi_{\nu} \) and \( \xi_{\nu} \geq n\gamma_{\nu} \), then the subspace iterates in (2.19a) satisfy \( \lim_{n \to \infty} \nabla C_n(\bar{L}[n]) = 0 \) almost surely; that is, \( Pr\{\lim_{n \to \infty} \nabla C_n(\bar{L}[n]) = 0\} = 1 \), and the subspace iterates will
converge to the stationary point of \((2.10)\).

**Proof:** The proof follows the steps in Proposition 2, with an extra step in the construction of the appropriate surrogate cost in Step 1. In particular, using that \(\forall n\) the optimal subspace is of the form \(\bar{L}_n = \Phi_n A\), the objective \(\tilde{f}_\nu\) can be further majorized over the subset of virtual subspaces \(\bar{L} = \Phi_n A\), by

\[
\tilde{f}_n(x_n; \Phi_n, A, q[n]) := f_n(x_n; \bar{L}[n-1], q[n]) + \text{tr}\{\nabla_L f_n(x_n; \bar{L}[n-1], q[n])(\Phi_n A - \bar{L}[n-1])^T\} + \frac{\xi_n}{2} \|A - \begin{bmatrix} A[n-1] \\ 0_{1 \times r} \end{bmatrix}\|_F^2
\]

for which we have

\[
\tilde{f}_n(x_n; \bar{L}, q[n]) - \tilde{f}_\nu(x_\nu; \Phi_n, A, q_\nu) = \frac{\gamma_n}{2} \|\bar{L} - \bar{L}[n-1]\|_{HS}^2 - \frac{\xi_n}{2} \|A - \begin{bmatrix} A[n-1] \\ 0_{1 \times r} \end{bmatrix}\|_F^2.
\]

The Cauchy-Schwarz inequality implies that

\[
\|\bar{L} - \bar{L}[n-1]\|_{HS}^2 = \|\Phi_n A - \Phi_n \begin{bmatrix} A[n-1] \\ 0_{1 \times r} \end{bmatrix}\|_{HS}^2 \leq \|\Phi_n\|_{HS}^2 \|A - \begin{bmatrix} A[n-1] \\ 0_{1 \times r} \end{bmatrix}\|_F^2
\]

and by choosing \(\xi_n \geq \|\Phi_n\|_F^2 \gamma_n = n \gamma_n\), we will have \(\tilde{f}_n(x_n; \bar{L}, q[n]) \leq \tilde{f}_\nu(x_\nu; \Phi_n, A, q_\nu)\).

Selecting now \(\tilde{f}_\nu(.\) as the new surrogate whose minimizer coincides with the update rule in \((2.19a)\), the rest of the proof follows that of Prop. 2.

\(\blacksquare\)

### 2.4 Reduced-complexity OK-FE on a budget

Per data vector processed, the iterative solvers of the previous section have one column of \(\Phi_n\) and one row of \(A\) added, which implies growing memory and complexity requirements as \(n\) grows. The present section combines two means of coping with this formidable challenge: one based on **censoring** uninformative data, and the second based on **budget maintenance**. By modifying
Algorithms 2 and 3 accordingly, memory and complexity requirements are rendered affordable.

2.4.1 Censoring uninformative data

In the LS cost that Algorithms 2 and 3 rely on, small values of the fitting error can be tolerated in practice without noticeable performance degradation. This suggests modifying the LS cost so that small fitting errors (say up to $\pm \epsilon$) induce no penalty, e.g., by invoking the $\epsilon$–insensitive cost that is popular in support vector regression (SVR) settings [53].

Consider henceforth positive-definite kernels for which low-rank factors offer an approximation to the full-rank kernel matrix, and lead to a generally nonzero LS-fit $\|\Phi_n - \bar{L}Q_n\|_2$. These considerations suggest replacing the LS cost

$$\ell(x_n; A[n - 1], q; x_{1:n-1})$$

with

$$\tilde{\ell}(x_n; A[n - 1], q; x_{1:n-1})$$

(2.23)

$$:= \begin{cases} 0 & \text{if } \ell(x_n; A[n - 1], q; x_{1:n-1}) < \epsilon \\ \ell(x_n; A[n - 1], q; x_{1:n-1}) - \epsilon & \text{otherwise} \end{cases}$$

By proper choice of $\epsilon$, the cost $\tilde{\ell}(.)$ implies that if $\ell(x_n; A[n - 1], q; x_{1:n-1}) < \epsilon$, the virtual $\phi(x_n)$ is captured well enough by the virtual current subspace $\bar{L}[n - 1] = \Phi_{n-1}A[n - 1]$, and the solver will not attempt to decrease its LS error, which suggests skipping the augmentation of $\Phi_{n-1}$, provided by the new lifted datum $\phi(x_n)$ [10].

In short, if the upper branch of (2.23) is in effect, $\phi(x_n)$ is deemed uninformative, and it is censored for the subspace update step; whereas having the lower branch deems $\phi(x_n)$ informative, and augments the basis set of the virtual subspace. The latter case gives rise to what we term online support vectors (OSV), which must be stored, while ‘censored’ data are discarded from subsequent subspace updates.

In order to keep track of the OSVs, let $S_{n-1}$ denote the set of indices corresponding to the SVs revealed up to time $n$. Accordingly, rewrite $\bar{L}[n - 1] = \Phi_{S_{n-1}}A[n - 1]$, and the modified LS cost as $\bar{\ell}(x_n; A[n - 1], q; x_{S_{n-1}})$, depending on which of the following two cases emerges.

C1. If $\bar{\ell}(x_n; A[n - 1], q; x_{S_{n-1}}) \leq \epsilon$, the OSV set will not grow, and we will have $S_n = S_{n-1}$; or

C2. If $\bar{\ell}(x_n; A[n - 1], q; x_{S_{n-1}}) > \epsilon$, the OSV set will grow, and we will have $S_n = S_{n-1} \cup \{n\}$. 
The subspace matrix per iteration will thus take the form \( \tilde{L}_{n} = \Phi_{S_n} A_{n} \), where \( \Phi_{S_n} := [\phi_{n_1}, ..., \phi_{n_{|S_n|}}] \), with \( S_n := \{n_1, n_2, ..., n_{|S_n|}\} \), and \( A \in \mathbb{R}^{|S_n| \times r} \). Upon replacing \( x_{1:n} \) in Algorithm 3 with \( x_{S_n} \), Algorithm 4 gives the pseudocode for our reduced-complexity online kernel-based feature extraction (OK-FE), which also includes a budget maintenance module that will be presented in the ensuing Section 2.4.2.

Modifying the LS-fit in (2.23) and discarding the censored data, certainly reduce the rate at which the memory and complexity requirements increase. In practice, thresholding is enforced after the budget is exceeded, when one needs to discard data. Regarding the selection of the threshold value, the later may be initialized at zero and be gradually increased until the desired censoring rate is reached (final threshold value will depend on the average fitting error and desired censoring rate); see also [10] for related issues. Albeit at a slower rate, \( |S_n| \) may still grow unbounded as time proceeds. Thus, one is motivated to restrict the number of OSVs to a prescribed affordable budget, \( |S_n| \leq B \), and introduce a solver which maintains such a budget throughout the iterations. To this end, we introduce next a greedy ‘budget maintenance’ scheme.

### 2.4.2 Budget maintenance

When inclusion of a new data vector into the OSV set pushes its cardinality \( |S_n| \) beyond the prescribed budget \( B \), the budget maintenance module will discard one SV from the SV set. The removal strategy is decided according to a predefined rule. In the following, we will describe two strategies for budget maintainence.

**Minimum-distortion removal rule**

In this scheme, the SV whose exclusion distorts the subspace \( \tilde{L}_{n} \) minimally will be discarded. Specifically, with \( \Phi_{n \setminus i} \) and \( A_{i \setminus [n]} \) denoting \( \Phi_{n} \) and \( A_{[n]} \) devoid of their \( i \)-th column and row, respectively, our rule for selecting the index to be excluded is

\[
i^* = \arg \min_{i \in S_n} \| \Phi_{n} A_{[n]} - \Phi_{n \setminus i} A_{i \setminus [n]} \|_{HS}^2
\]

\[
= \arg \min_{i \in S_n} \text{tr} \{ A^T_{i \setminus [n]} K(x_{S_n \setminus i}, x_{S_n}) A_{[n]} - 2 A^T_{i \setminus [n]} K(x_{S_n \setminus i}, x_{S_n}) A_{i \setminus [n]} + A^T_{i \setminus [n]} K(x_{S_n \setminus i}, x_{S_n \setminus i}) A_{i \setminus [n]} \}.
\] (2.24)
Enumeration over $S_n$ and evaluation of the cost incurs complexity $O(B^3)$ for solving (2.24). Hence, in order to mitigate the computational complexity, a greedy scheme is put forth. Since exclusion of an SV will result in removing the corresponding row from the subspace factor, discarding the SV corresponding to the row with the smallest $\ell_2-$norm suggests a reasonable heuristic greedy policy. To this end, one needs to find the index

$$\hat{i}_* = \arg\min_{i=1,2,...,B+1} \|a_i[n]\|_2$$

where $a_i^\top [n]$ denotes the $i-$th row of $A[n]$. Subsequently, $\hat{i}_*$ as well as the corresponding SV are discarded from $S_n$ and the SV set respectively, and an OSV set of cardinality $|S_n| = B$ is maintained.

**Remark 3.** In principle, methods related to those in [128], including replacement of two SVs by a linear combination of the two, or projecting an SV on the SV set and discarding the projected SV, are also viable alternatives. In practice however, their improved performance relative to (2.25) is negligible and along with their increased complexity, renders such alternatives less attractive for large-scale datasets.

**Recency-aware removal rule**

This policy is tailored for tracking applications, where the subspace capturing the data vectors can change dynamically. As the subspace evolves, the fitting error will gradually increase, indicating the gap between the true and learned subspace, thus requiring incorporation of new vectors into the subspace. In order for the algorithm to track a dynamic subspace on a fixed budget, the budget maintenance module must gradually discard outdated SVs inherited from “old” subspaces, and include new SVs. Therefore, apart from “goodness-of-fit” (cf. (2.25)), any policy tailored to tracking should also take into account “recency” when deciding which SV is to be discarded.

To this end, corresponding to the $i-$th SV, let $\eta_i$ denote the recency factor whose value is initialized to 1. For every inclusion of a new SV, the recency $\eta_i$ of the current SVs will be degraded by a forgetting factor $0 < \beta \leq 1$; that is, $\eta_i$ will be replaced by $\beta \eta_i$. Consequently, older SVs will have smaller $\eta_i$ value whereas recent vectors will have $\eta_i \simeq 1$. To incorporate this memory factor into the budget maintenance module, our idea is to choose the SV to be discarded
according to
\[ \hat{i}_* = \arg \min_{i=1,2,\ldots,B+1} \eta_i \| a_i[n] \|_2 \]  

(2.26)

which promotes discarding older SVs over more recent ones.

By tuning \( \beta \), the proposed memory-aware budget maintenance module can cover a range of different schemes. For large values of \( \beta \approx 1 \), it follows that \( \eta_i \approx \eta_j \forall i, j \in \mathcal{S} \), and (2.26) approaches the minimum distortion removal rule in (2.25), which is tailored for learning static subspaces. On the other hand, for small \( \beta \), the discarding rule is heavily biased towards removing old SVs rather than the newly-included ones, thus pushing the maintenance strategy towards a first-in-first-out (FIFO) approach, which is often optimal for applications with fast-varying subspaces. Algorithms 4 and 5 tabulate the updates and the greedy budget maintenance scheme, respectively. Budget maintenance strategy in (2.25) is a special case of Alg. 5 with \( \beta = 1 \).

2.4.3 Complexity analysis

Computational complexity of the proposed OK-FEB is evaluated in the present section. The computations required by the \( n \)-th iteration of Alg. 4 for feature extraction and parameter update depend on \( B, r, \) and \( D \), as well as the censoring process outlined in Section 2.4.1. Specifically, computing \( \mathcal{G}_n \) and performing the first-order stochastic update that yields \( A[n] \) requires \( \mathcal{O}(B^2r) \) multiplications, a cost that is saved for skipped instances when \( \ell_n < \epsilon \). Regarding the computation of \( q[n] \), \( Br(B+r) \) multiplications are needed to form \( A^\top[n-1]K(x_{S_{n-1}}, x_{S_{n-1}})A[n-1] \), and \( \mathcal{O}(r^3) \) multiplications for the inversion of \( A^\top[n-1]K(x_{S_{n-1}}, x_{S_{n-1}})A[n-1] + \lambda I_r \). Fortunately, the aforementioned computations can also be avoided for iteration \( n \), if the previous iteration performs no update on \( A[n-1] \); in this case, \( (A^\top[n-1]K(x_{S_{n-1}}, x_{S_{n-1}})A[n-1] + \lambda I_r)^{-1} \) remains unchanged and can simply be accessed from memory. Nevertheless, a “baseline” of computations is required for feature extraction related operations that take place regardless of censoring. Indeed, forming \( A^\top[n-1]k(x_{S_{n-1}}, x_n) \) requires \( Br \) multiplications for the matrix-vector product, and \( \mathcal{O}(BD) \) for the evaluation of \( B \) kernels in \( k(x_{S_{n-1}}, x_n) \); the matrix-vector product that remains for obtaining \( q[n] \) requires \( r^2 \) additional multiplications.

Overall, running OK-FEB on \( N \) data and with a value of \( \epsilon \) such that \( \tilde{N} \leq N \) data are used for updates requires \( \mathcal{O}(\tilde{N}(Br(B+r) + r^3) + N(B(D+r) + r^2)) \). Alternatively, tuning \( \epsilon \) such that \( \Pr\{\ell_n > \epsilon\} = \mathbb{E}[\tilde{N}/N] := \rho \) yields an expected complexity \( \mathcal{O}(N(Br(\rho(B+r) + 1) + (\rho r + 1)r^2 + BD)) \). As simulation tests will corroborate, the budget parameter \( B \) can
Algorithm 4 Online Kernel-based Feature Extraction on a Budget (OKFEB)

Input \(\{x_\nu\}_{\nu=1}^n, \lambda\)
Initialize \(A[1]\) at random and \(S_1 = \{1\}\)
For \(n = 2, \ldots\) do

\[ q[n] = \left( (A^\top[n-1]K(x_{S_{n-1}}, x_{S_{n-1}})A[n-1] + \lambda I_r)^{-1} \times A^\top[n-1]k(x_{S_{n-1}}, x_n) \right. \]

\[ \ell_n = k(x_n, x_n) - 2k^\top(x_{S_{n-1}}, x_n)A[n-1]q[n] \]

\[ + q^\top[n]A^\top[n-1]K(x_{S_{n-1}}, x_{S_{n-1}})A[n-1]q[n] \]

if \(\ell_n < \epsilon\) then
\(S_n = S_{n-1}\)
else
\(S_n = S_{n-1} \cup \{n\}\)
\(\hat{G}_n = K(x_{S_n}, x_{S_n}) \left[ A[n-1] \begin{bmatrix} 0_r \times 1 \end{bmatrix} \right] q[n]q^\top[n] \]
\[- k(x_{S_n}, x_{S_n})q^\top[n] + \frac{\lambda}{n} K(x_{S_n}, x_{S_n}) \left[ A[n-1] \begin{bmatrix} 0_r \times 1 \end{bmatrix} \right] \]
\(A[n] = \begin{bmatrix} A[n-1] \begin{bmatrix} 0_r \times 1 \end{bmatrix} \end{bmatrix} - \mu_{n,A} \hat{G}_n \)

if \(|S_n| > B\) then
Run budget maintenance module
end if
end if
EndFor
Return \(A[n], S_n, \{q[\nu]\}_{\nu=1}^n\)

Algorithm 5 Budget maintenance module

Input \(\{S, A, \{\eta_i\}_{i \in S}\}\)
\(\eta_i \leftarrow \beta \eta_i \quad \forall i \in S\)
\(i_s = \arg \min_{i \in S} \eta_i \|a_i^\top\|_2\)
\(S \leftarrow S \setminus \{i_s\}\)
Discard the \(i_s\)-th row of \(A\) and \(\eta_s\)
Return \(\{S, A, \{\eta_i\}_{i \in S}\}\)
be chosen as $B = cr$ with $c \in [1.5, 5]$. Thus, we can simplify the overall complexity order as $O(Nr^2(\rho r + 1) + NDt)$.  

### 2.5 Stability of kernel approximation

In this section, the effect of low-rank approximation of the lifted vectors on kernel-matrix approximation as well as kernel-based classification and regression is analytically quantified. Recall that given $\{x_\nu\}_{\nu=1}^N$, the virtual subspace obtained by running OK-FEB is $\tilde{\mathbf{L}} = \Phi_S \mathbf{A} \in \mathbb{R}^{D \times r}$, and the corresponding projection coefficients are $\mathbf{Q}_N$. By defining the random variables $e_i := \|\phi(x_i) - \hat{\phi}(x_i)\|_2^2 = \|\phi(x_i) - \tilde{\mathbf{L}}q_i\|_2^2$ capturing the LS error, we have the following result.

**Proposition 4.** If the random variables $e_i \in [0, 1]$ are i.i.d. with mean $\bar{e} := \mathbb{E}[e_i]$, then for kernels satisfying $|\kappa(x_i, x_j)| \leq 1$, the matrix $\mathbf{K} = \Phi^\top \Phi$ can be approximated by $\hat{\mathbf{K}} := \hat{\Phi}^\top \hat{\Phi}$, and with probability at least $1 - 2e^{-2Nt^2}$, it holds that

$$\frac{1}{N} \|\mathbf{K} - \hat{\mathbf{K}}\|_F \leq \sqrt{\bar{e} + t} (\sqrt{\bar{e} + t} + 2).$$

*(2.27)*

**Proof:** Upon defining $\tilde{\mathbf{E}} := \hat{\Phi} - \hat{\Phi}$, one can write

$$\|\mathbf{K} - \hat{\mathbf{K}}\|_F = \|\Phi^\top \Phi - \hat{\Phi}^\top \hat{\Phi}\|_F$$

$$= \|\Phi^\top \Phi - (\Phi + \tilde{\mathbf{E}})^\top (\Phi + \tilde{\mathbf{E}})\|_F$$

$$= \|2\tilde{\mathbf{E}}^\top \Phi + \tilde{\mathbf{E}}^\top \tilde{\mathbf{E}}\|_F$$

$$\leq 2\|\tilde{\mathbf{E}}\|_{HS}\|\Phi\|_{HS} + \|\tilde{\mathbf{E}}\|_{HS}^2$$

*(2.28a)*

$$\leq 2\sqrt{N}\|\tilde{\mathbf{E}}\|_{HS} + \|\tilde{\mathbf{E}}\|_{HS}^2$$

*(2.28b)*

where in *(2.28a)* we used the triangle inequality for the Frobenious norm along with the property $\|BC\|_F \leq \|B\|_F \|C\|_F$, and *(2.28b)* holds because for, e.g., radial kernels satisfying $|\kappa(x_i, x_j)| \leq 1$, we have

$$\|\Phi\|_{HS} := \sqrt{\text{tr}(\Phi^\top \Phi)} = \sqrt{\sum_{i=1}^N \kappa(x_i, x_i)} \leq \sqrt{N}.$$
Furthermore, since $\|\bar{E}\|_F := \sqrt{\sum_{i=1}^{N} e_i}$, and $\bar{e}_N := (1/N) \sum_{i=1}^{N} e_i$ with $e_i \in [0, 1]$, Hoeffding’s inequality yields $\Pr\left(\bar{e}_N - \bar{e} \geq t\right) \leq e^{-2Nt^2}$, which in turn implies

$$\Pr\left(\frac{1}{N}\|\bar{E}\|_F^2 \geq \bar{e} + t\right) = \Pr\left(\bar{e}_N \geq \bar{e} + t\right) \leq e^{-2Nt^2}.$$  \hspace{1cm} (2.29)

Finally, taking into account (2.28b), it follows that with probability at least $1 - 2e^{-2Nt^2}$, we have

$$\|K - \hat{K}\|_F \leq N(2\sqrt{\bar{e}} + t + (\bar{e} + t)).$$  \hspace{1cm} (2.30)

Proposition 4 essentially bounds the kernel approximation mismatch based on how well the projection onto the subspace approximates the lifted data $\phi(x)$.

**Remark 4.** Consider now decomposing the kernel matrix as

$$\hat{K} := \Phi^\top \hat{\Phi} = (LQ)^\top (LQ) = Q^\top A^\top \Phi_S^\top \Phi_S A Q = Q^\top A^\top K_S A Q = Z^\top Z$$  \hspace{1cm} (2.31)

where matrix $Z := K_S^{1/2} A Q$ has size $|S| \times N$, and $S$ denotes the budgeted SV set. This factorization of $\hat{K}$ could have resulted from a linear kernel over the $|S| \times 1$ training data vectors forming the $N$ columns of $Z$. Thus, for kernel-based tasks such as kernel classification, regression, and clustering applied to large datasets, we can simply map the $D \times N$ data $X$ to the corresponding features $Z$ trained via the proposed solvers, and then simply rely on fast linear learning methods to approximate the original kernel-based learning task; that is to approximate the function $f(x) = \sum_{i \in S} c_i \kappa(x, x_i)$ by the linear function $g(z) = w^\top z$ expressed via the extracted features. Since linear pattern recognition tasks incur complexity $O(NB^2)$, they scale extremely well for large datasets (with $N \gg$), compared to kernel SVM that incurs complexity $O(N^3)$. Furthermore, in the testing phase, evaluation of function $f(x)$ requires $\kappa(x_v, x_i)$ for $\forall i \in S$ to be evaluated at complexity $O(|S|D)$, where $|S|$ is the number of SVs that typically grows with $N$. In contrast, if approximated by the linear $g(z)$, function evaluation requires $O(BD + Br)$ operations including the feature extraction and function evaluation. Setting the budget $B$ to 1.5 to 5 times the rank parameter $r$, our complexity is of order $O(rD + r^2)$, which represents a considerable decrease over $O(|S|D)$.

Subsequently, we wish to quantify how the performance of linear classification and regression
based on the features $K^{1/2}_{S}AQ$ compares to the one obtained when training with the exact kernel matrix $K$.

### 2.5.1 Stability analysis for kernel-based classification

Kernel-based SVM classifiers solve [104, p. 205]

$$
\alpha^* = \arg \min_{\alpha} \frac{1}{2} \alpha^T YKY \alpha - 1^T \alpha \\
\text{s.t. } y^T \alpha = 0, \ 0 \leq \alpha \leq \frac{C}{N} 1_N
$$

where $Y$ is the diagonal matrix with the $i$-th label $y_i$ as its $i$-th diagonal entry, $y^T := [y_1, y_2, \ldots, y_N]$, and $1_N$ is an $n \times 1$ vector of 1’s. Solution (2.32) corresponds to the dual variables of the primal optimization problem, which yields

$$
\hat{w}^* = \arg \min_{\bar{w} \in \mathbb{R}^D} \frac{1}{2} \| \bar{w} \|_H^2 + \frac{C}{N} \sum_{i=1}^N \max \{0, 1 - y_i \bar{w}^T \phi(x_i)\}.
$$

(2.33)

Here, parameter $C$ controls the trade-off between maximization of the margin $1/\|w\|_H$, and minimization of the misclassification penalty, while the solution of (2.33) can be expressed as $\hat{w}^* = \sum_{i=1}^N \alpha_i^* y_i \phi(x_i)$. [104] p.187.

Exploiting the reduced memory requirement offered through the low-rank approximation of the kernel matrix via OK-FEB, the dual problem can be approximated as

$$
\hat{\alpha}^* = \arg \min_{\alpha} \frac{1}{2} \alpha^T \hat{K} \alpha - 1^T \alpha \\
\text{s.t. } y^T \alpha = 0, \ 0 \leq \alpha \leq \frac{C}{N} 1_N.
$$

(2.34)

Viewing $\hat{K}$ as a linear kernel matrix over $\{\hat{\phi}(x_i)\}$s (cf. Remark 4), similar to (2.32), the minimization (2.34) can be re-written in the primal form as

$$
\hat{w}^* = \arg \min_{\bar{w}} \frac{1}{2} \| \bar{w} \|_H^2 + \frac{C}{N} \sum_{i=1}^N \max \{0, 1 - y_i \bar{w}^T \hat{\phi}(x_i)\}
$$

(2.35)

for which we have $\hat{w}^* = \sum_{i=1}^N \hat{\alpha}_i^* y_i \hat{\phi}(x_i)$. Upon defining the random variable $\xi_i := \|\phi(x_i) - \hat{\phi}(x_i)\|_H$ with expected value $\hat{\xi} := \mathbb{E}[\xi_i]$, the following proposition quantifies the gap between
\( \hat{w}^* \) and \( \tilde{w}^* \).

**Proposition 5.** If \( \xi_i \in [0, 1] \) are i.i.d., with mean \( \bar{\xi} \), the mismatch between the linear classifiers given by (2.33) and (2.35) can be bounded, and with probability at least \( 1 - e^{-2Nt^2} \), we have

\[
\| \Delta w \|_H^2 := \| \tilde{w}^* - \hat{w}^* \|_H^2 \leq 2C^{3/2}\left( \bar{\xi} + t \right).
\] (2.36)

**Proof:** It clearly holds that

\[
\| \Delta w \|_H^2 \leq \frac{C}{N} (\| \tilde{w}^* \|_H + \| \hat{w}^* \|_H) \sum_{i=1}^{N} \| \phi(x_i) - \hat{\phi}(x_i) \|_H
\]

\[
\leq \frac{2C^{3/2}}{N} \sum_{i=1}^{N} \| \phi(x_i) - \hat{\phi}(x_i) \|_H \leq 2C^{3/2}(\bar{\xi} + t)
\]

where the first inequality relies on the strong convexity of (2.33), (2.35), and the fact that \( \| \tilde{w}^* \|_H \leq \sqrt{C} \) and \( \| \hat{w}^* \|_H \leq \sqrt{C} \) [106]; while the second inequality holds with probability at least \( 1 - e^{-2Nt^2} \) using Hoeffding’s inequality for \( \bar{\xi}_N := (1/N) \sum_{i=1}^{N} \xi_i \).

Note that under the i.i.d. assumption on \( e_i := \| \phi(x_i) - \hat{\phi}(x_i) \|_H \), random variables \( \xi_i \) are also i.i.d., rendering the conditions of Propositions 4 and 5 equivalent.

Next, we study the performance of linear SVMs trained on the set \( \{z_i, y_i\}_{i=1}^{N} \), where \( z_i := \frac{K_{1/2}}{S}Aq_i \); that is, the linear function \( g(z) = w^Tz \) is learned by finding

\[
w^* = \arg \min_{w \in \mathbb{R}^r} \frac{1}{2} \|w\|^2 + \frac{C}{N} \sum_{i=1}^{N} \max\{0, 1 - y_i w^Tz_i\}.
\] (2.37)

The following result asserts that the classifiers learned through (2.35) and (2.37) can afford identical generalization capabilities.

**Proposition 6.** The generalization capability of classifiers (2.35) and (2.37) is identical, in the sense that \( \hat{w}^* \hat{T} \hat{\phi}(x) = w^* T z \).

**Proof:** Since for the low-rank approximation of the kernel matrix we have \( \hat{K} = Z^T Z \), then (2.34) and (2.37) are equivalent, and consequently \( w^* = \sum_{i=1}^{N} \hat{\alpha}_i^* y_i z_i \). Now, one can further expand \( \hat{w}^* \hat{T} \hat{\phi}(x) \) and \( w^* T z \) to obtain

\[
\hat{w}^* \hat{T} \hat{\phi}(x) = \sum_{i=1}^{N} \hat{\alpha}_i^* y_i \hat{T} (x_i) \hat{\phi}(x) = \sum_{i=1}^{N} \hat{\alpha}_i^* y_i q_i^T A^T \hat{\Phi}_S \hat{\Phi}_S A q
\]


\[
\begin{align*}
\text{and } w^T z &= \sum \alpha_i^* y_i \mathbf{z}_i^T \mathbf{z} \\
&= \sum \alpha_i^* y_i \mathbf{q}_i^T \mathbf{A}^T \hat{\Phi}_S^T \hat{\Phi}_S \mathbf{A} \mathbf{q}
\end{align*}
\]

where the equivalence follows readily.

In addition to markedly reduced computational cost when utilizing linear (L)SVM, our novel classifier can also be efficiently trained online \([106]\) as new data becomes available (or iteratively when the entire datasets cannot be stored in memory which necessitates one-by-one acquisition). In this case, the proposed OK-FEB in Algorithm 4 can be run in parallel with the online classifier training, an attribute most suitable for big data applications.

### 2.5.2 Stability analysis for kernel-based regression

Consider now the kernel-based ridge regression task on the dataset \(\{x_i, y_i\}_{i=1}^N\), namely

\[
\min_{\beta} \frac{1}{N} \|y - K\beta\|_2^2 + \lambda \beta^T K \beta
\]

which admits the closed-form solution \(\beta^* = (K + \lambda N I)^{-1} y\) \([104]\), p. 251]. Alleviating the \(O(N^2)\) memory requirement through low-rank approximation of matrix \(K\), the kernel-based ridge regression can be approximated by

\[
\min_{\beta} \frac{1}{N} \|y - \hat{K}\beta\|_2^2 + \lambda \beta^T \hat{K} \beta
\]

whose solution is given as \(\hat{\beta}^* = (\hat{K} + \lambda N I)^{-1} y\). The following proposition bounds the mismatch between \(\beta^*\) and \(\hat{\beta}^*\).

**Proposition 7.** If the random variables \(e_i \in [0, 1]\) are i.i.d., with mean \(\overline{e}\), and \(|y_i| \leq B_y\) for \(i = 1, 2, \ldots, N\), with probability at least \(1 - 2e^{-2Nt^2}\), we have

\[
\|\beta^* - \hat{\beta}^*\|^2 \leq \frac{B_y}{\lambda^2} \sqrt{\overline{e} + t} (\sqrt{\overline{e} + t} + 2).
\]

**Proof:** Following \([19]\), we can write

\[
\beta^* - \hat{\beta}^* = (K + \lambda N I)^{-1} y - (\hat{K} + \lambda N I)^{-1} y
\]

\[
= - \left( (K + \lambda N I)^{-1} (K - \hat{K}) (K + \lambda N I)^{-1} \right) y
\]

where we have used the identity \(P^{-1} - \hat{P}^{-1} = -P^{-1}(\hat{P} - P)\hat{P}^{-1}\), which holds for any invertible matrices \(P\) and \(\hat{P}\). Taking the \(\ell_2\)-norm of both sides and using the Cauchy-Schwartz...
inequality, we arrive at
\[
\| \beta^* - \hat{\beta}^* \| \leq \| (K + \lambda N I)^{-1} \| \| K - \hat{K} \| \| (\hat{K} + \lambda N I)^{-1} \| \| y \|
\]
\[
\leq \frac{\| K - \hat{K} \| N B_y}{\lambda_{\min}(K + \lambda N I) \lambda_{\min}(\hat{K} + \lambda N I)}
\]
\[
\leq \frac{B_y \| K - \hat{K} \|_2}{\lambda^2 N}.
\]

(2.41)

Using the inequality \( \| P \|_2 \leq \| P \|_F \) along with Proposition 4, yields the bound with probability \( 1 - 2e^{-2Nt^2} \).

\[ \blacksquare \]

2.6 Numerical tests

This section presents numerical evaluation of various performance metrics to test our proposed algorithms using both synthetic and real datasets. In subsection 2.6.1 we empirically study the proposed batch and online feature extraction algorithms using a toy synthetic dataset. In subsection 2.6.2 we focus on the tracking capability of the proposed OK-FEB and demonstrate its performance in terms of the evolution of average LS-fitting error obtained at iteration \( n \) as \( (1/n) \sum_{\nu=1}^{n} \| \phi(x_\nu) - \bar{L}[n] q_\nu \|_H^2 \). Regarding the kernel matrix approximation performance, given a window size \( N_{\text{wind}} \), we have \( (N - N_{\text{wind}}) \) windows in a dataset of size \( N \). Consequently, the mismatch of kernel matrix approximation is averaged over all such windows, and it is thus obtained as
\[
\frac{1}{N - N_{\text{wind}}} \sum_{w=1}^{N - N_{\text{wind}}} \left( \frac{1}{N_{\text{wind}}} \| K_w - \hat{K}_w \|_F \right)
\]

where \( K_w \) and \( \hat{K}_w \) are the kernel matrix and its approximation over the data vectors in the \( w \)-th window. Finally, in subsection 2.6.3 we test how well OK-FEB approximates the kernel-based classification and regression modules, and compare its performance with competing alternatives.

2.6.1 Kernel-based feature extraction: Batch vs. online

Performance of Algorithms 1, 2 and 3 on solving the minimization (2.10) is tested using synthetically generated data arriving in streaming mode with \( \nu = 1, 2, \ldots, 5,000 \). The test involves generating two equiprobable classes of \( 3 \times 1 \) data vectors \( \{x_\nu\} \), each uniformly drawn from the surface of a sphere centered at the origin with radius \( R_{c1} = 1 \) or \( R_{c2} = 2 \), depending
on whether its label $y_\nu$ equals 1 or $-1$, respectively. Noise drawn from the Gaussian distribution $\mathcal{N}(\mathbf{0}_{3 \times 1}, \sigma^2 \mathbf{I}_{3 \times 3})$ is added to each $x_\nu$, with $\sigma^2$ controlling the overlap between the two classes. Linear classifiers cannot correctly classify data generated in this manner. For this reason, the Gaussian kernel $\kappa(x_i, x_j) = \exp(-\|x_i - x_j\|^2_2/\gamma)$ was used with $\gamma = 100$. The online schemes can solve the problem on-the-fly, while the batch Algorithm 1 is also employed to solve (2.10) offline. We compare the overall LS fit given by the subspace update $\bar{L}[n]$ using the three different solvers across time (iteration) index $n$. The parameters for the OK-FE solvers are chosen as $\mu_{n,L} \propto 1/n$, $\mu_{n,A} \propto 1/n^2$, $\lambda = 10^{-3}$, and the maximum number of iterations in the batch solver is set to $I_{\text{max}} = 50$.

Figure 2.1(a) depicts how stochastic low-complexity updates of $A$ in the online solvers ensure convergence of the average LS cost to the high-complexity batch solution for $r = 7$. When $n$ is small, the low-rank approximation is accurate and the resulting LS error in Batch-KFE is small. Note however that LS is nonzero for $n < r$, due to regularization. As $n$ increases, the number of vectors in the batch minimization also increases, while $r$ is fixed. Thus, the fitting problem becomes more challenging and the LS error increases slightly until $n$ is large enough and the $n$ data vectors are representative of the pdf from which data is drawn - a case that the LS fit stabilizes. Fig. 2.1(b) plots the convergence curve for Algs. 2 and 3.

While the Gaussian kernel that was adopted here is the most widely used type, other kernels are also applicable (e.g. polynomial kernels). Although it goes beyond the scope and claims of this paper, similar to all kernel-based schemes, the effect of not knowing the ideal kernel can be mitigated via data-driven multi-kernel approaches [6]. Plotted in Fig. 2.2 is the fitting error for different kernels with different parameters versus $r$ to highlight this issue (in Gaussian kernel, $\gamma = 2\sigma^2$).

In addition, Fig. 2.3 plots the evolution of the average LS cost across iterations for different choices of parameters $(r, B)$ in the OK-FEB solver. Note that relative to the batch Alg. 1 that incurs complexity $O(N^2 r)$ per iteration, OK-FE exhibits similar performance at much lower complexity $O(Nr^3 +NDr)$.

### 2.6.2 Dynamic subspace tracking

In this subsection, we assess efficiency of the novel approach in tracking dynamic subspaces using synthetic and real-world datasets.
Synthetic data

We generated a set of $N = 2,000$ data vectors in $\mathbb{R}^3$. For $n = 1, \ldots, 1000$ the data were drawn from the surface of the sphere given by the manifold $(x_1/3)^2 + x_2^2 + x_3^2 = 1$, while for $n = 1,001, \ldots, 2,000$ they were sampled from the surface of the spheroid $x_1^2 + (x_2/3)^2 + (x_3)^2 = 1$, in Fig. 2.4. Plotted in Fig. 2.5 is the LS error of the low-rank feature extraction with $r = 10$ and kernel parameter $\gamma = 2$ (averaged by a window of length 200 for improved visualization) across time $n$. To enable tracking, the step size at sample index $\nu$ is chosen as $\mu_\nu = 1/\|q_\nu\|_2$. As the plot suggests, the change of the manifold at $n = 1,000$ can be spotted by the rise in the LS error. The tracking capability of OK-FEB enables the subspace to adapt to the change in the underlying manifold. However, within a window of fixed subspace, namely for $1 < n < 1,000$, and $1,200 < n < 2,000$, and especially for small budget $B = 2r$, the budget maintenance...
Figure 2.2: LS-fit of OKFE for different choices of polynomial and Gaussian kernels with different parameters.

Figure 2.3: LS-fit for different choices of \((r, B)\) using OK-FEB policy in Alg. 3 outperforms the FIFO budget maintenance policy by carefully discarding the SVs whose exclusion least distorts the learned subspace. Among the budgeted algorithms, setting small \(\beta\) leads to a forceful exclusion of relatively older vectors, and thus adaptation to the new subspace at \(t = 1000\) takes place faster. In contrast, having small \(\beta\) reduces the capability of fine tuning to the underlying subspace when it is not changing. This is corroborated by the lower curve for \(\beta = 0.9\) versus \(\beta = 1\) during the subspace change, while \(\beta = 1\) gives lower error when subspace is not changing. Budget size \(B = 2r\) demonstrates such effects more clearly as smaller \(B\) requires a more careful selection of the support vectors, hence emphasizing the effect of parameter \(\beta\). The performance of the batch solver with no budget size constraint is
Figure 2.4: Visualization of the nonlinear synthetic manifolds

Figure 2.5: LS-fitting error of dynamic dataset versus time

also plotted, whose average fitting error is worse than that of budget size $B = 5r$ and is similar to the very restrictive budget size $B = 2r$. This is contributed to the fact that in the batch solver, the union of two subspaces is approximated by low-rank $r$, and thus the performance is inferior to the proposed online approach which is capable of tracking the underlying subspace. Overall, given the dynamics of a particular dataset, selection of $\beta$ directly sets the operation mode of our subspace learning, and is tunable to the pace of dynamics.

Average mismatch of $\hat{K}$ found from OK-FEB for various values of rank $r$ and choice of $B = 2r$ is plotted in Fig. 2.6, and is compared with KPCA as well as state-of-the-art variations of the Nystrom approximation, namely Improved Nystrom [143], SS-Nystrom [127], and MEKA [111]. Considering the dynamic nature of the data, the mismatch is evaluated over a moving window of length $N_{wind} = 100$, and averaged over all such windows. As the plot suggests,
OK-FEB outperforms competing alternatives and better suites datasets with dynamic subspaces.

2.6.3 Real-data on physical activity tracking

In this subsection, we test performance of OK-FEB on the physical activity monitoring dataset PAMAP2 [101]. The dataset contains $N = 129,200$ measurements from 3 Colibri wireless inertial measurement units (MU) worn by 9 subjects during different physical activities, such as walking and cycling. The MUs are placed on the dominant arm, chest, and dominant ankle of the subjects, each recording 13 quantities including acceleration and gyroscope data with sampling frequency 100Hz. We discarded all measurement vectors with missing entries, idle state measurements, and first and last 1,000 measurements of each activity, as they correspond to transient states. The tests are performed on data corresponding to subject number 1, and can be similarly repeated for other subjects as well.

The data is fed to OK-FEB with $(r, B) = (10, 15)$, and step size set to $\mu_t = 1/\|q_t\|_2$. LS error given by the nonlinear feature extraction (averaged over a window of length 200 for improved visualization) is plotted in Fig. 2.7 across time. Every activity is also coded to a number in $(0, 1]$, and plotted in the same figure versus time to highlight the activity changes over time. As the figure illustrates, different activities correspond to different manifolds, each of which can be approximated with a certain accuracy via dynamic subspace learning and feature extraction. Introducing the forgetting factor $\beta < 1$ enhances the learning capability. Table 2.1 reports the average LS-error and its variance for different activities using various budget maintenance strategies, with $\beta = 1, 0.9$, and the FIFO strategy.

Similar to Fig. 2.6, Fig. 2.8 depicts the average mismatch of kernel matrix approximation of OK-FEB with $B = 1.5r$ for the PAMAP2 dataset. Comparison with the competing Nystrom
Table 2.1: Mean and variance of LS-fitting error of the extracted features with \((r, B) = (10, 15)\) for different activities using different budget maintenance strategies

<table>
<thead>
<tr>
<th>Code</th>
<th>Activity</th>
<th>(\beta = 1)</th>
<th>(\beta = 0.9)</th>
<th>FIFO Bud.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3</td>
<td>Walking</td>
<td>0.099</td>
<td>0.074</td>
<td>0.074</td>
</tr>
<tr>
<td></td>
<td></td>
<td>±0.016</td>
<td>±0.012</td>
<td>±0.012</td>
</tr>
<tr>
<td>0.4</td>
<td>Running</td>
<td>0.227</td>
<td>0.187</td>
<td>0.187</td>
</tr>
<tr>
<td></td>
<td></td>
<td>±0.025</td>
<td>±0.022</td>
<td>±0.022</td>
</tr>
<tr>
<td>0.5</td>
<td>Cycling</td>
<td>0.058</td>
<td>0.028</td>
<td>0.028</td>
</tr>
<tr>
<td></td>
<td></td>
<td>±0.027</td>
<td>±0.012</td>
<td>±0.12</td>
</tr>
<tr>
<td>0.6</td>
<td>Nordic Walking</td>
<td>0.130</td>
<td>0.103</td>
<td>0.103</td>
</tr>
<tr>
<td></td>
<td></td>
<td>±0.020</td>
<td>±0.016</td>
<td>±0.016</td>
</tr>
<tr>
<td>0.7</td>
<td>Ascending Stairs</td>
<td>0.079</td>
<td>0.063</td>
<td>0.063</td>
</tr>
<tr>
<td></td>
<td></td>
<td>±0.022</td>
<td>±0.018</td>
<td>±0.018</td>
</tr>
<tr>
<td>0.8</td>
<td>Descending Stairs</td>
<td>0.094</td>
<td>0.066</td>
<td>0.065</td>
</tr>
<tr>
<td></td>
<td></td>
<td>±0.021</td>
<td>±0.016</td>
<td>±0.016</td>
</tr>
<tr>
<td>0.9</td>
<td>Vacuum cleaning</td>
<td>0.045</td>
<td>0.029</td>
<td>0.029</td>
</tr>
<tr>
<td></td>
<td></td>
<td>±0.013</td>
<td>±0.008</td>
<td>±0.008</td>
</tr>
<tr>
<td>1.0</td>
<td>Rope jumping</td>
<td>0.272</td>
<td>0.238</td>
<td>0.238</td>
</tr>
<tr>
<td></td>
<td></td>
<td>±0.063</td>
<td>±0.057</td>
<td>±0.057</td>
</tr>
</tbody>
</table>

Figure 2.7: LS-fitting error of the PAMAP2 dataset versus time
variations in [143] and [111] clearly demonstrates the advantage of OK-FEB with forgetting factor $\beta = 0.9$. Due to the large number of data vectors, KPCA and SS-Nystrom could not be implemented.

2.6.4 Online regression and classification

In this subsection, the generalization capability of the online linear classification and regression modules based on the features $Z$ returned by OK-FEB is tested. We compare the performance of linear regression and classification as well as competing online kernel-based learners including (unbudgeted) Perceptron [38], (unbudgeted) Norma [63], (unbudgeted) online gradient descent (OGD) [106], (unbudgeted) online dictionary learning (ODL), and budgeted online gradient descent (BOGD) [128], Forgetron [22], Projectron [88], and budgeted passive-aggressive algorithm (BPA) [130] with our novel OK-FEB, where the acquired features $z_n$ are fed to online linear [106] and regularized-LMS solvers for the classification and regression tasks, respectively. The size and specifications of the dataset used are listed in Table 2.2 and are accessible from the LIBSVM website [http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/] or the UCI machine learning repository [http://www.ics.uci.edu/~mlearn/]. The parameter values used per dataset are reported in Table 2.2. In particular, tuning of the Frobenious norm regularization and kernel bandwidth parameters are done via cross validation over a discretized grid. Regarding the budget, to ensure stability of the algorithm it suffices that we set $B > r$, while it has been observed that setting $B$ very high yields only marginal improvement in terms of accuracy. Finally, for the selection of $r$, we test an increasing sequence of values starting from $r = 2$ and gradually increasing until the improvement in terms of fitting error becomes negligible. The aforementioned process is typically used to determine the minimum required complexity of
Table 2.2: Specifications of datasets.

<table>
<thead>
<tr>
<th>dataset</th>
<th>$D$</th>
<th>$N$</th>
<th>$r$</th>
<th>$B$</th>
<th>$\gamma$</th>
<th>$C$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adult</td>
<td>123</td>
<td>32K</td>
<td>50</td>
<td>1.2$r$</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>CADATA</td>
<td>8</td>
<td>20.6K</td>
<td>5</td>
<td>1.5$r$</td>
<td>$7 \times 10^7$</td>
<td>0.01</td>
</tr>
<tr>
<td>Slice</td>
<td>384</td>
<td>53.5K</td>
<td>10</td>
<td>1.2$r$</td>
<td>50</td>
<td>0.01</td>
</tr>
<tr>
<td>Year</td>
<td>90</td>
<td>463.7K</td>
<td>10</td>
<td>1.2$r$</td>
<td>$5 \times 10^7$</td>
<td>0.01</td>
</tr>
</tbody>
</table>

Parametric models (e.g., order-adaptive least-squares [60]). The censoring threshold $\epsilon$ is set using a moving-average of LS-error values for the past 100 data vectors.

Classification and regression accuracy as well as run time are plotted versus iteration index. Perceptron, Norma, ODL, and OGD are unbudgeted algorithms, and their SV sets (dictionary atoms in ODL) grow as iteration index increases in Fig. 2.9. Although the accuracy of these algorithms can serve as a benchmark, their run time grows the fastest. Thus, for the “Year” dataset ($N \gg$), the mentioned algorithms are run only over 10% of the data vectors. As these tests demonstrate, among the budgeted algorithms, OK-FEB reliably approximates the kernel function through the extracted features, thus offering more accurate classification and regression performance when compared to existing alternatives.
Figure 2.9: Online classification tests on (a) Adult, and regression tests on (b) CADATA, (c) Slice, and (d) Year datasets.
Chapter 3

Overlapping Community Identification via Constrained Egonet Tensor Decomposition

3.1 Preliminaries and the top-down approach

Given a network of \( N \) vertices (or nodes) \( v \in \mathcal{V} \) where \(|\mathcal{V}| = N\), and their edgeset \( \mathcal{E} \), community detection aims at finding subsets of nodes, a.k.a. clusters or communities, for which resident nodes demonstrate dense intra-community connections while distinct communities are sparsely connected. A cover is defined as the set of such communities, with “desirable covers” exhibiting certain characteristics, namely: i) constituent communities should include dense intra-connections and sparse inter-connections; ii) communities of very large sizes are not appealing as they bear little information on the underlying structure of the network; and, iii) the union of the identified communities should cover the entire graph, leaving few or no “homeless” nodes, not assigned to any community.

The proposed method, called “DC-EgoTen,” relies on the construction of an egonet-based multi-dimensional representation of the network. It utilizes “EgoTen” to solve a sequence of nonnegative tensor decomposition subproblems, and progressively unveils the identified communities over the graph. Let us treat EgoTen as a black-box module in this section, postponing its detailed explanation to Section 3.2 and further delineate the overall algorithm here.
In particular, DC-EgoTen takes a top-down approach for the overall task of community identification. To this end, “EgoTen” is initially applied over the entire network to provide an assignment of nodes to a few “coarse” communities. Each of the detected communities is in fact a subset of nodes, inducing a subgraph in the overall graph. Thus, the identified “coarse” communities are further amenable to a subsequent application of EgoTen for unraveling a more refined community structure. This procedure can be applied consecutively for a number of times over each of the detected communities, creating a tree of communities, until the desired resolution, i.e., maximum acceptable community size, is achieved for all detected communities (at the leaves of the tree). In Section 3.2, the proposed egonet-based multi-dimensional graph representation is introduced, and “EgoTen” as our core toolbox for community detection is detailed.

3.2 Egonet-tensor construction and constrained decomposition

Given graph \( G = (V, E) \), the binary adjacency matrix \( W \in \mathbb{R}^{N \times N} \) is constructed by setting the \((i, j)\)-th entry as \( w_{ij} = 1 \) if \((i, j) \in E\), and \( w_{ij} = 0 \), otherwise. Furthermore, the egonet of node \( n \) is defined as the subgraph induced by node \( n \), its one-hop neighbors denoted by \( \mathcal{N}(n) \), and all their connections \[^{[2]}\]. Thus, the egonet of node \( n \) can be conveniently represented by the induced subgraph \( G^{(n)} := (V, E^{(n)}) \), where \( E^{(n)} \) is the edge set of the links in between nodes \( \{n\} \cup \mathcal{N}(n) \). Subsequently, the egonet adjacency matrix \( W^{(n)} \in \mathbb{R}^{N \times N} \) is defined as

\[
 w^{(n)}_{ij} := \begin{cases} 
 w_{ij} & \text{if } (i, j) \in E^{(n)} \\
 0 & \text{otherwise.}
\end{cases}
\]

Typically, the center node \( n \) is excluded from \( G^{(n)} \), but it is included here for convenience.

Let us now consider a three-way egonet-tensor \( \mathbf{W} \in \mathbb{R}^{N \times N \times N} \) constructed by contanetating egonet adjacency matrices \( W^{(n)} \) for all nodes \( n \in V \) in the frontal slabs of \( \mathbf{W} \). In tensor parlance, that is tantamount to setting the \( n \)-th frontal slab of \( \mathbf{W} \) as \( \mathbf{W}_{:, :, n} := W^{(n)} \), where \( : \) is a free index that spans its range.

The advantage of representing a graph via its egonet-tensor is due to the fact that tensors as multi-way data structures are capable of capturing higher-order connectivities, namely two-hop links among neighboring nodes. Thus, in networks where overlapping as well as highly-mixed
communities render the task of community detection very challenging, egonet-tensors provide a rich representation of the graph, which will be leveraged in the upcoming algorithm. The egonet-based representation is also of interest particularly in the absence of extra nodal features, as the enhanced representation is a result of careful exploitation of the adjacency matrix where no other source of information is provided.

Fig. 3.1 illustrates the egonet-tensor construction procedure, while Algorithm 6 provides its pseudocode. In the ensuing subsection we cast the task of community detection as a constrained tensor decomposition over the egonet-tensor $\mathbf{W}$, elaborate on the intuition behind the proposed approach, and introduce EgoTen as its efficient solver.

3.2.1 EgoTen: A constrained tensor decomposition approach

In order to gain insights into the properties of the introduced egonet-tensor, consider the toy network whose connectivity is depicted in Figure 3.2.a. The network under consideration comprises five communities with dense intra-community and fewer inter-community connections. Upon constructing the egonet-tensor and after permutation (so that resident nodes are indexed right after one another), it becomes evident that the egonet-tensor demonstrates a block structure;
Figure 3.2: (a) A toy network with 5 non-overlapping communities; (b) corresponding egonet-tensor; and (c) its community-revealing factorization via PARAFAC decomposition.

see Fig. 3.2.b. In particular, dense diagonal blocks in the tensor capture the dense intra-community links, while spare off-diagonal entries represent inter-community connections.

Had the communities been complete sub-graphs, each block would have been an all-one three-way tensor (considering non-zero diagonal entries provided by self-loops), which could have been readily decomposed into the outer product of three all-one vectors (each of the size of the community); that is, $1_{p \times p \times p} = 1_{p \times 1} \odot 1_{p \times 1} \odot 1_{p \times 1}$, where $p$ is the size of the community. Moreover, had the communities been disjoint, that is if no inter-community links were present, the egonet-tensor could have been readily written as the summation of five tensors, each of whom can be effectively approximated by the outer-product of three vectors; see Fig. 3.2.c.

Such decomposition is indeed reminiscent of the well-known canonical polyadic decomposition (CPD) [64] also known as PARAFAC, where the number of terms, i.e., the rank of the decomposition, reveals the number of communities. Prompted by this observation, let us introduce the constrained nonnegative PARAFAC over the egonet adjacency tensor $W$ as

$$\{ \hat{A}, \hat{B}, \hat{C} \} = \arg \min_{A,B,C} \left\{ \|W - \sum_{k=1}^{K} a_k \odot b_k \odot c_k\|_F^2 + \lambda (\|A\|_F^2 + \|B\|_F^2) \right\} \quad (3.1)$$

$$\text{s.t.} \quad A \geq 0, B \geq 0, C \geq 0$$

$$\sum_{k=1}^{K} c_{nk} = 1 \quad \forall n = 1, 2, ..., N$$

The first term in the objective is the original Frobenious term in the well-known PARAFAC, through which minimization of the mismatch between the multi-way data structure $W$ and its approximation is achieved. Furthermore, nonnegativity of the egonet-tensor is effected
through additional constraints over the factors $A := [a_1, \ldots, a_K]$, $B := [b_1, \ldots, b_K]$ and $C := [c_1, \ldots, c_K]$. Regarding the simplex constraints on the rows of matrix $C$, let us now focus on the $n$-th frontal slab of the egonet-tensor. One can readily show that the tensor approximation gives rise to the following decomposition

$$W^{(n)} \simeq \sum_{k=1}^{K} c_{nk}(a_k \circ b_k)$$

(3.2)

where $c_{nk}$ denotes the $(n, k)$-th entry of factor $C$. As stated earlier, parameter $K$ is referred to as the rank of the decomposition, and in this application reveals the number of identified communities. Thus, such decomposition can be interpreted as a weighted sum over $K$ “basis” $\{a_k \circ b_k\}^K_{k=1}$, where $(a_k \circ b_k)$ captures the “connectivity structure” within the $k$-th community. Consequently, $c_{nk}$ can be viewed as association level of node $n$ to community $k$. Thus, the simplex constraint over the rows of matrix $C$ readily guarantees a normalized association vector for every node in the graph to the identified $K$ communities. Finally, the Frobenious regularizers over factors $A$ and $B$ simply resolve the scaling ambiguity between the two factors, and is different from [8].

The overall optimization in (3.1) is a trilinear block-convex problem [59], whose solver is detailed in the following subsection.

### 3.2.2 Constrained PARAFAC solver

Exploiting the block-convex structure of the constrained PARAFAC in (3.1), the optimization can be solved by alternating minimization, where each of $A$, $B$, $C$ is optimized respectively by fixing the other two at their current values. Factors are repeatedly updated until a stopping criterion or a maximum number of iterations is achieved. Considering iteration $i$, factors are updated as follows.

**Factor $A$ update**

Fixing factors $B^{(i-1)}$ and $C^{(i-1)}$ at their current values, the update of factor $A$ is obtained by the corresponding subproblem, which after algebraic manipulation can be readily rewritten as a regularized nonnegative least-squares (LS) minimization as
\[
A^{(i)} = \arg \min_{\mathbf{A} \geq 0} \| \mathbf{W}_1 - \mathbf{H}_A^{(i)} \mathbf{A}^\top \|_F^2 + \lambda \| \mathbf{A} \|_F^2
\]  
(3.3)

where \( \mathbf{W}_1 := [\text{vec} (\mathbf{W}_{1^{(i-1)}}, \ldots, \text{vec} (\mathbf{W}_{N^{(i-1)}}))] \in \mathbb{R}^{N^2 \times N} \) is a matricized reshaping of the tensor \( \mathbf{W} \), and \( \mathbf{H}_A^{(i)} := \left[ b_1^{(i-1)} \otimes c_1^{(i-1)}, \ldots, b_K^{(i-1)} \otimes c_K^{(i-1)} \right] \), with \( b_c^{(i-1)} (c_c^{(i-1)}) \) denoting column \( c \) of \( \mathbf{B}^{(i-1)} \) (resp. \( \mathbf{C}^{(i-1)} \)), and \( \otimes \) the Kronecker product operator; see also [64]. Solving the subproblem in (3.4) by the alternating direction method of multipliers (ADMM), the augmented Laugrangian of the cost is

\[
L_A^{(i)}(\mathbf{A}, \bar{\mathbf{A}}, \mathbf{Y}) = \| \mathbf{W}_1 - \mathbf{H}_A^{(i)} \bar{\mathbf{A}}^\top \|_F^2 + \lambda \| \bar{\mathbf{A}} \|_F^2 + r_+(\mathbf{A}) + (\rho/2) \| \mathbf{Y} + \mathbf{A} - \bar{\mathbf{A}} \|_F^2
\]  
(3.4)

where \( \bar{\mathbf{A}}, \mathbf{Y} \in \mathbb{R}^{N \times K} \) are the auxiliary and dual variables, respectively, and \( r_+(\mathbf{A}) \) is the regularizer corresponding to the nonnegativity constraint,

\[
r_+(\mathbf{A}) := \begin{cases} 0 & \text{if } \mathbf{A} \geq 0 \\ +\infty & \text{o.w.} \end{cases}
\]

Simulated tests suggest that selection of the regularization parameter \( \rho = \| \mathbf{H}_A^{(i)} \|_F^2 / K \) can provide near-optimal performance [59], and that is the choice adopted henceforth.

The ADMM solver then proceeds by iteratively updating blocks of variables \( \mathbf{A}, \bar{\mathbf{A}}, \mathbf{Y} \) as

\[
\begin{align*}
\bar{\mathbf{A}}^{(r)} & = \arg \min_{\bar{\mathbf{A}}} L_A^{(i)}(\mathbf{A}^{(r-1)}, \bar{\mathbf{A}}, \mathbf{Y}^{(r-1)}) \\
& = \left( \mathbf{H}_A^{(i)} \mathbf{H}_A^{(i)} \mathbf{I}_K \times K + (\lambda + \rho/2) \mathbf{I}_K \times K \right)^{-1} \\
& \times \left( \mathbf{W}_1 \mathbf{H}_A^{(i)} + \frac{\rho}{2} (\mathbf{Y}^{(r-1)} + \mathbf{A}^{(r-1)}) \right) \\
\mathbf{A}^{(r)} & = \mathcal{P}_+(\mathbf{Y}^{(r-1)} - \bar{\mathbf{A}}^{(r)}) \\
\mathbf{Y}^{(r)} & = \mathbf{Y}^{(r-1)} - \rho (\mathbf{A}^{(r)} - \bar{\mathbf{A}}^{(r)}) \\
r & = r + 1
\end{align*}
\]

(3.5)

until \( \| \mathbf{A}^{(r)} - \mathbf{A}^{(r-1)} \| / \| \mathbf{A}^{(r-1)} \| \leq \epsilon \), or the maximum number of iterations is exceeded. Upon its termination, factor \( \mathbf{A} \) is updated as \( \mathbf{A}^{(i)} \leftarrow \mathbf{A}^{(r)} \), and the algorithm proceeds with updating factor \( \mathbf{B} \) as in the following.
Factor B update

Upon fixing $A = A^{(i)}$ and $C = C^{(i-1)}$, factor $B$ is updated by solving the subproblem

$$B^{(i)} = \arg \min_{B \geq 0} \| W_2 - H_B^{(i)} B^\top \|_F^2 + \lambda \| B \|_F^2$$  \hspace{1cm} (3.6)

where $W_2 := [\text{vec}(W_{1,1}^{(i)}), \ldots, \text{vec}(W_{1,N}^{(i)})] \in \mathbb{R}^{N^2 \times N}$, and $H_B^{(i)} := [a_1^{(i)} \otimes c_1^{(i-1)}, \ldots, a_K^{(i)} \otimes c_K^{(i-1)}]$, yielding a similar optimization problem as in (3.4).

Undertaking the same approach as for (3.5), the ADMM update for solving (3.6) yields

$$\begin{cases}  
  B^{(r)} &= \left( H_B^{(i)} H_B^{(i)} + (\lambda + \rho/2)I_{K \times K} \right)^{-1} \\
  &\times \left( W_2 H_B^{(i)} + \frac{\rho}{2} (Y^{(r-1)} + B^{(r-1)}) \right) \\
  \bar{B}^{(r)} &= P_+ \left( Y^{(r-1)} - \bar{B}^{(r)} \right) \\
  B^{(r)} &= \mathcal{P}_+ \left( Y^{(r-1)} - \rho (B^{(r)} - \bar{B}^{(r)}) \right) \\
  Y^{(r)} &= Y^{(r-1)} - \rho (B^{(r)} - \bar{B}^{(r)}) \\
  r &= r + 1.
\end{cases}$$  \hspace{1cm} (3.7)

Upon the termination of (3.7) due to either attaining the stopping criterion or reaching the maximum number of iterations, factor $B$ is updated as $B^{(i)} \leftarrow B^{(r)}$.

Factor C update

Fixing factors $A = A^{(i)}$ and $B = B^{(i)}$, update of factor $C$ is obtained by solving the subproblem

$$C^{(i)} = \arg \min_C \| W_3 - H_C^{(i)} C^\top \|_F^2$$  \hspace{1cm} (3.8)

s.t. \hspace{0.5cm} C \geq 0, \sum_{k=1}^K c_{nk} = 1 \quad \forall n = 1, \ldots, N

where $W_3 := [\text{vec}(W_{1,1}^{(i)}), \ldots, \text{vec}(W_{1,N}^{(i)})]$ is the matricized version of $\mathbf{W}$ along the 3-rd mode, and $H_C^{(i)} := [a_1^{(i)} \otimes b_1^{(i)}, \ldots, a_K^{(i)} \otimes b_K^{(i)}]$. Augmented Lagrangian of the cost can be readily formed as

$$\mathcal{L}_C^{(i)}(C, \bar{C}, Y) = \| W_3 - H_C^{(i)} C^\top \|_F^2 + \rho_\text{simp}(C)$$

$$+ \rho/2 \| Y + C - \bar{C} \|_F^2.$$
where \( r_{\text{simp}}(C) \) is the regularizer corresponding to the simplex constraint on the rows of matrix \( C \) as

\[
  r_{\text{simp}}(C) := \begin{cases} 
    0 & \text{if } C \geq 0, \sum_{k=1}^{K} c_{nk} = 1 \forall n \\
    +\infty & \text{o.w.}
  \end{cases}
\]

The ADMM solver then proceeds by iteratively updating the blocks of variables \( C, \bar{C}, Y \) as

\[
  \begin{align*}
    \bar{C}(r) &= \arg \min_{\bar{C}} L_{C}^{(i)}(C^{(r-1)}, \bar{C}, Y^{(r-1)}) \\
    &= \left( H_{C}^{(i)\top} H_{C}^{(i)} + \rho/2 I_{K \times K} \right)^{-1} \\
    & \times \left( W_{3}^{\top} H_{C}^{(i)} + \frac{\rho}{2} (Y^{(r-1)} + C^{(r-1)}) \right) \\
    C^{(r)} &= \mathcal{P}_{\text{simp}}(Y^{(r-1)} - \bar{C}(r)) \\
    Y^{(r)} &= Y^{(r-1)} - \rho (C^{(r)} - \bar{C}(r)) \\
    r &= r + 1.
  \end{align*}
\]  

Projection of the rows of matrix \((Y^{(r-1)} - \bar{C}(r))\) onto the simplex set can be achieved via the algorithm in [30]. Upon termination, factor \( C \) is updated as \( C^{(i)} \leftarrow C^{(r)} \).

Once the overall trilinear optimization in (3.1) is solved, factor \( C \) unravels soft community association of the nodes. Extraction of hard communities based on the learned PARAFAC model is discussed in the next section. Also, Algorithm 7 lists the pseudocode of the proposed EgoTen followed by hard community assignments.

### 3.3 Community assignment and quality evaluation

As discussed in Section 3.2, the introduced EgoTen community detection algorithm aims at solving a constrained decomposition of the egonet-tensor, thus providing factor \( C \) whose entries unravel soft community associations. In order to transform the “soft” to “hard” memberships, one can simply utilize a threshold approach, according to which if \( c_{nk} > \tau_k \), node \( n \) is assigned to community \( k \), and it is not assigned otherwise. The main challenge here is on selecting a proper threshold \( \tau_k \). To this end, let \( \hat{C}_k \) denote the set of nodes in community \( k \) (with hard
Algorithm 7 EgoTen Community Detection Core Algorithm

**procedure** {\textsc{Egoden}}(\(W, K\))

Initialize \(A, B, C \in \mathbb{R}^{N \times K}\) at random and set \(i = 0\)

while \(i < I_{\text{max}}\) do or not-converged

\(A^{(i)} \leftarrow\) Solve (3.4) using (3.5)

\(B^{(i)} \leftarrow\) Solve (3.6) using (3.7)

\(C^{(i)} \leftarrow\) Solve (3.8) using (3.9)

\(i \leftarrow i + 1\)

end while

for \(k = 1, 2, \ldots, K\) do

\(\hat{C}_k = \{}\)

for \(n = 1, 2, \ldots, N\) do \(\hat{C}_k \leftarrow \hat{C}_k \cup \{n\}\) if \(c_{nk} \geq \tau_k\)

end for

end for

end procedure

return \(\{\hat{C}_k\}_{k=1}^{K}\)

memberships), and define its conductance as [37]

\[
\phi(\hat{C}_k) := \frac{\sum_{i \in \hat{C}_k, j \notin \hat{C}_k} W_{ij}}{\min\{\text{vol}(\hat{C}_k), \text{vol}(V \setminus \hat{C}_k)\}}
\]

where

\[
\text{vol}(\hat{C}_k) := \sum_{i \in \hat{C}_k, \forall j} W_{ij}
\]

and \((V \setminus \hat{C}_k)\) is the complement of \(\hat{C}_k\). According to \(\phi(.)\), high-quality communities yield small conductance scores as they exhibit dense connections among the nodes within the community and sparse connections with the rest.

Considering conductance as a measure of community quality, we can now set threshold \(\tau_k\) such that the quality of community \(k\) after hard member assignment is maximized. In order to lower complexity, we simply choose \(\tau_k\) from the discretized range \([1/K, 2/K, \ldots]\). Note that having an association level \(c_{nk} = 1/K\) \(\forall k\) for a given node \(n\) is tantamount to having an equally favorable association with the \(K\) communities, and having threshold \(\tau_k = 1/K\) will result in a community assignment if the association is higher than this uniform level. Also, setting \(\tau_k = 1/K\) together with the simplex constraints on the rows of factor \(C\) guarantees that every node will be assigned to at least one community, and no node will be left unassigned.
However, tuning $\tau_k$ to obtain low conductance communities improves quality.

### 3.3.1 DC-EgoTen

Having delineated different modules of DC-EgoTen, we are ready to present the overall algorithm. Given graph $G = (V, E)$, DC-EgoTen initially constructs the egonet-tensor $\mathbf{W}$ using Alg. 6, applies EgoTen in Alg. 7 over $\mathbf{W}$, and obtains detected communities $\{\hat{C}_k\}_{k=1}^K$. Next, the resolution of $\hat{C}_k$ for $k = 1, 2, \ldots$ will determine whether further refining is necessary for each of the identified communities. That is, if $|\hat{C}_k| < C_{\text{max}}$, the resolution of detected community $\hat{C}_k$ is satisfactory, and no further processing is required. On the other hand, if $|\hat{C}_k| > C_{\text{max}}$, the subgraph induced by the set of nodes in $\hat{C}_k$ will be extracted, over which the entire process will be repeated. Algorithm 8 lists the pseudocode for the overall DC-EgoTen.

Figure 3.3 provides a schematic over our toy network with five communities, each of size $|C_k| = 15$ for $k = 1, 2, \ldots, 5$. In this example, in every EgoTen the rank parameter is $K = 2$, which gives rise to a binary tree of detected communities. As in this example, in the first application of EgoTen, the green community is detected by the constrained PARAFAC, while the rest of the network is ‘lumped’ together in the second community. Thus, the green community needs no further processing as its size is below $C_{\text{max}} = 20$, while application of EgoTen on the second term gives rise to two relatively more refined communities. Proceeding with another set of EgoTen application on the detected communities will reveal the remaining clusters, creating overall five leaves in the tree, corresponding to the detected fine-resolution communities.

![Figure 3.3: The proposed DC-EgoTen community detection algorithm on a toy example.](image)
Algorithm 8 DC-EgoTen

procedure DC-EgoTen($\mathcal{V}$, $\mathbf{W}$)
    Set parameters $K, C_{\text{max}}$
    Define global cover set $\mathcal{S} = \{\}$
    $\mathbf{W} \leftarrow$ Egonet-tensor construction($\mathcal{V}$, $\mathbf{W}$)
    $\{C_i\}_{i=1,2,...,K} \leftarrow$ EgoTen($\mathbf{W}$, $K$)
    for $C \in \{C_i\}_{i=1,2,...,K}$ do
        # If community $C$ is refined enough, add it to the cover set $\mathcal{S}$, otherwise refine it using EgoTen
        if $|C| < C_{\text{max}}$ then
            $\mathcal{S} \leftarrow \mathcal{S} \cup C$
        else
            # Extract the subgraph of nodes in $C$
            $\mathbf{W}_{\text{sub}} \leftarrow \text{subgraph}(C, \mathbf{W})$
            DC-EgoTen ($C$, $\mathbf{W}_{\text{sub}}$)
        end if
    end for
end procedure

return $\mathcal{S}$

If an oracle had provided the number of underlying communities, the algorithm would have identified all clusters in its first application of EgoTen by setting $K = 5$. However, successive application of EgoTen with smaller target rank $K$ can compensate for the lack of such information, which is almost-always encountered in practice. Furthermore, DC-EgoTen nicely proceeds with the desiderata of community identification discussed in Section 3.1 because: i) the multi-dimensional egonet-based representation captures multi-hop connectivities, leading to an improved quality in the detected communities; ii) consecutive division of large communities enhances resolution; and, iii) setting threshold parameter $\tau_k = 1/K$ in EgoTen can guarantee a full coverage of the network, while its tuning can further control the trade-off between coverage and quality.

3.3.2 Performance evaluation

In addition to conductance, normalized mutual information and F1-score are measures for assessing the performance of community identification when ground-truth communities are provided.

Normalized mutual information (NMI) [35]: Given $\mathcal{S}^* = \{C_1^*, \ldots, C_{|S|}^*\}$ and $\hat{\mathcal{S}} = \{\hat{C}_1, \ldots, \hat{C}_{|\hat{S}|}\}$
as ground-truth and detected covers, respectively, the information theoretic measure NMI is defined as (cf. [35])

$$\text{NMI}(S^*, \hat{S}) := \frac{2I(S^*, \hat{S})}{H(S^*) + H(\hat{S})}$$

where $H(\hat{S})$ denotes the entropy of set $\hat{S}$ defined as

$$H(\hat{S}) := -\sum_{i=1}^{\hat{S}} p(\hat{C}_i) \log p(\hat{C}_i) = -\sum_{i=1}^{\hat{S}} \frac{|\hat{C}_i|}{N} \log \frac{|\hat{C}_i|}{N}$$

and similarly for $H(S^*)$. Furthermore, $I(S^*, \hat{S})$ denotes the mutual information between $S^*$ and $\hat{S}$, defined as

$$I(S^*, \hat{S}) := \sum_{i=1}^{S^*} \sum_{j=1}^{\hat{S}} \frac{|C_i^* \cap \hat{C}_j|}{N} \log \frac{N|C_i^* \cap \hat{C}_j|}{|C_i^*||\hat{C}_j|}.$$ (3.10)

Intuitively, the mutual information $I(S^*, \hat{S})$ reflects a measure of similarity between the two covers. Thus, high values of NMI, namely its maximum at 1, reflect high accuracy in community identification, whereas low values of NMI, namely its minimum at 0, represent poor discovery of the true underlying communities. This measure has been generalized for overlapping communities in [68], and will be utilized for performance assessment in such networks.

**Average F1-score** [136]: F1-score is a measure of binary classification accuracy. Specifically, the harmonic mean of precision and recall takes its highest value at 1 and lowest value at 0. Average F1-score for detected cover $\hat{S}$ is

$$\bar{F}_1 := \frac{1}{2|S^*|} \sum_{i=1}^{S^*} F1(C_i^*, \hat{C}_{I(i)}) + \frac{1}{2|\hat{S}|} \sum_{i=1}^{\hat{S}} F1(C_i^* , \hat{C}_{I'(i)})$$

where

$$I(i) = \arg \max_j F1(C_i^*, \hat{C}_j), \quad I'(i) = \arg \max_j F1(C_j^*, \hat{C}_i)$$

in which $F1(C, C_j) := \frac{2|C \cap C_j|}{|C| + |C_j|}$. 
3.4 Numerical tests

In this section, the proposed DC-EgoTen is applied to synthetic as well as real datasets. Synthetic Lancichinetti-Fortunato-and-Radicci (LFR) networks [69] are utilized as a benchmark to study the resilience and performance of different community identification algorithms in the presence of overlapping as well as mixing communities.

3.4.1 LFR benchmark networks

LFR graphs serve as benchmark networks in which certain real-world properties, namely power-law distribution for nodal degree and community sizes, as well as the presence of overlapping and mixing communities are preserved. Such networks are configured by a total number of $N$ nodes, $\bar{d}$ average degree, and power-law distribution exponents $\gamma_1$ and $\gamma_2$ for degree and community sizes, respectively. Furthermore, parameter $\mu$ controls the community mixing, where higher values result in more out-of-community edges in between non-resident nodes. Moreover, parameters $o_n$, $o_m$ respectively set the number of overlapping nodes and communities (with which these nodes are associated).

In order to assess the resilience of the proposed DC-EgoTen to variations of $\mu$ and $o_n$, we have generated networks with $N = 2,000$, $\bar{d} = 100$, $\gamma_1 = 2$, $\gamma_2 = 1$, and varied $\mu \in [0.1, 0.7]$ as well as $o_n$ in $10\% - 70\%$ of the total networks size $N$, respectively. DC-EgoTen is run by setting the rank $K$ in the initial application as $K = 100$, while following applications are set as $K = 2$, essentially leading to a bisection of the network in the subsequent steps, and sparse tensor decompositions are handled via the SPLATT toolbox [117]. Thresholding parameter $\tau_k$ is selected as explained in Section 3.3 for the top EgoTen (allowing for overlapping community detection), and set as $\tau = 1/2$ for next steps. Maximum community size is set as $C_{max} = 200$. The performance is compared with state-of-the-art algorithms BigClam [136], Demon [20], and Nise [132] with ‘spread-hub’ seeding strategy, where $|\hat{S}| = 200$ is provided as an estimate on the number of communities in Nise and BigClam. Due to the availability of underlying communities, the performance is assessed via NMI and F1-scores and averaged over 10 realizations of the network for each setting.

As the results in Figures 3.4 and 3.5 corroborate, DC-EgoTen provides higher performance in terms of NMI and F1-score, thanks to the rich egonet-based representation as well as the progressive identification of refined communities.
Figure 3.4: Performance of different algorithms versus different community mixing values $\mu$ for $o_n = 600$, and $o_m = 3$.

Figure 3.5: Performance of different algorithms versus different number of overlapping nodes $o_n$ for $\mu = 0.2$, and $o_m = 3$.

3.4.2 Real-world networks

In this subsection, the performance of DC-EgoTen is compared with state-of-the-art overlapping community detection algorithms on various real-world networks, listed in Table [3.1] available in [71]. In DC-EgoTen, constructing the egonet-tensors as well as solving the constrained PARAFAC utilize parallel implementation, while Bigclam and Nise also allow for parallel threading. Thus, for networks with $N < 1$ million, these algorithms are run using 8 threads and 32GB of RAM, while for the Youtube dataset, 24 threads with 256 GB of RAM are utilized. As with synthetic datasets, we apply DC-EgoTen with $K = 100$ for the first application of EgoTen, and set $K = 2$ for subsequent steps. Threshold parameter $\tau_k$ is selected as explained in Section
\[3.3\] for the top EgoTen (allowing for overlapping community detection), and set as \(\tau = 1/2\) for next steps. Also, maximum community size \(C_{\text{max}}\) is set to 1\% of the network size for each dataset.

Table 3.1: Real-world networks.

| Dataset     | No. of vertices \(N\) | No. of edges \(|E|\) | Edge type   |
|-------------|------------------------|----------------------|-------------|
| Facebook    | 4,039                  | 88,234               | Undirected  |
| Enron       | 36,692                 | 183,831              | Undirected  |
| Epinion     | 75,879                 | 508,837              | Directed    |
| Slashdot    | 82,168                 | 948,464              | Directed    |
| Email       | 265,214                | 420,045              | Directed    |
| Stanford    | 281,903                | 2,312,497            | Directed    |
| Notre dame  | 325,729                | 1,497,134            | Directed    |
| Youtube     | 1,134,890              | 2,987,624            | Undirected  |

Figure 3.6 plots the run time of different algorithms while Table 3.2 lists the coverage and number of detected communities. Due to unavailability of ground-truth communities, NMI and F1-score could not be evaluated, thus performance is assessed using the conductance-coverage curve. To this end, for a given algorithm, the conductance of the identified communities is computed and the communities are sorted accordingly in an increasing order. Conductance-coverage curve is then plotted by increasing the maximum conductance, and progressively adding the sorted communities to the set of covered nodes.

Figure 3.7 depicts the aforementioned curve for various datasets. As low values of conductance correspond to more cohesive communities, a smaller area under curve (AUC) generally implies better performance. However, the resolution of the communities is another important metric which must be considered in drawing conclusions. Interestingly, the separation of different scattered points for a given algorithm in the conductance-coverage curve reveals the granularity of the detected communities. That is, if a detected community is very large, its inclusion creates a jump in the coverage, which is noticeable by the two consecutive points in the plot being placed far apart. Thus, examining Figure 3.7 reveals that the identified communities via DC-EgoTen and Bigclam are usually of more refined sizes as those plots are always smooth, while the performance of Nise and Demon is often limited to detecting very large communities (upto 40\% of the whole network). Furthermore, although one may not particularly be interested in 100\% coverage, it is desirable that a relatively high number of nodes to be covered within the detected communities, and thus low coverage where more than 50\% of the nodes are left uncovered is
Figure 3.6: Runtime of different algorithms on various datasets denoted on the x-axis as: (D1) Facebook, (D2) Enron, (D3) Epinion, (D4) Email, (D5) Slashdot, (D6) Notredame, (D7) Stanford, and (D8) Youtube.

considered undesirable.
Table 3.2: Coverage and number of detected communities of different methods over real-world networks.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>DC-EgoTen</th>
<th>Bigclam</th>
<th>Demon</th>
<th>Nise</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Coverage</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Facebook</td>
<td>100%</td>
<td>95%</td>
<td>99%</td>
<td>89%</td>
</tr>
<tr>
<td></td>
<td>No. of comm.</td>
<td>523</td>
<td>500</td>
<td>8</td>
</tr>
<tr>
<td>Enron</td>
<td>100%</td>
<td>90%</td>
<td>65%</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>No. of comm.</td>
<td>553</td>
<td>500</td>
<td>343</td>
</tr>
<tr>
<td>Slashdot</td>
<td>100%</td>
<td>100%</td>
<td>95%</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>No. of comm.</td>
<td>1163</td>
<td>500</td>
<td>51</td>
</tr>
<tr>
<td>Epinion</td>
<td>100%</td>
<td>100%</td>
<td>35%</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>No. of comm.</td>
<td>1274</td>
<td>2000</td>
<td>136</td>
</tr>
<tr>
<td>Email</td>
<td>100%</td>
<td>83%</td>
<td>11%</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>No. of comm.</td>
<td>965</td>
<td>2000</td>
<td>24</td>
</tr>
<tr>
<td>Notredame</td>
<td>100%</td>
<td>100%</td>
<td>39%</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>No. of comm.</td>
<td>1169</td>
<td>2000</td>
<td>1497</td>
</tr>
<tr>
<td>Stanford</td>
<td>100%</td>
<td>90%</td>
<td>85%</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>No. of comm.</td>
<td>807</td>
<td>2000</td>
<td>2596</td>
</tr>
<tr>
<td>Youtube</td>
<td>100%</td>
<td>100%</td>
<td>22%</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>No. of comm.</td>
<td>813</td>
<td>5000</td>
<td>3835</td>
</tr>
</tbody>
</table>
Figure 3.7: Conductance-coverage curve for various datasets using different community detection algorithms.
Chapter 4

Minimum Uncertainty Based Detection of Adversaries in Deep Neural Networks

4.1 Bayesian neural network preliminaries

Bayesian inference is among the powerful tools utilized for analytically understanding and quantifying uncertainty in DNNs [118, 39]. In this section, we provide a short review on the basics of Bayesian neural networks, and move on to the inference phase for adversary detection in Section 4.1.2 which is of primary interest in this work.

Consider an \( L \)-layer deep neural network, which maps the input \( x \in \mathcal{X} \) to output \( y \in \mathcal{Y} \). The weights are denoted by \( \omega := \{W_i\}_{i=1}^L \), and are modeled as random variables with prior probability density function (pdf) \( p(\omega) \).

Given training input \( X := [x_1, x_2, ..., x_n] \) and output data \( Y := [y_1, y_2, ..., y_n] \), it is assumed that the parameters \( \omega \) only depend on these \((X, Y)\) data. As a result, the predictive pdf for a new input \( x_\nu \) can be obtained via marginalization as [40]

\[
p(y_\nu | x_\nu, X, Y) = \int p(y_\nu | x_\nu, \omega)p(\omega | X, Y)d\omega
\]  

which requires knowing the conditional \( p(\omega | X, Y) \). The complexity of estimating \( p(\omega | X, Y) \) motivates well the variational inference (VI) approach, where \( p(\omega | X, Y) \) is replaced by a
surrogate pdf $q_\theta(\omega)$ that is parameterized by $\theta$. For $q_\theta(\omega)$, it is desired to: (D1) approximate closely $p(\omega|X, Y)$; and, (D2) provide easy marginalization in (4.1) either in closed form or empirically. To meet (D1), the surrogate is chosen by minimizing the Kullback-Leibler (KL) divergence $KL(p(\omega|X, Y), q_\theta(\omega))$, which is subsequently approximated by the log evidence lower bound \[12, p. 462\]

$$L_{VI}(\theta) := \int q_\theta(\omega) \log p(Y|X, \omega) \, d\omega - KL(q_\theta(\omega)||p(\omega)). \quad (4.2)$$

Finding $q_\theta$ boils down to maximizing the log evidence lower bound, that is, $\theta_{VI} = \arg \max_\theta L_{VI}(\theta)$. A common choice for $q_\theta(\omega)$ to also satisfy (D2) is described next.

### 4.1.1 Variational inference

A simple yet effective choice for $q_\theta(\omega)$ is a factored form modeling the weights as independent across layers, that is

$$q_\theta(\omega) = \prod_{l=1}^{L} q(W_l; M_l, \theta_z) \quad (4.3)$$

where the $l$-th layer with $h_l$ hidden units is modeled as

$$W_l = M_l \text{diag}([z_{l,1}, z_{l,2}, \ldots, z_{l,h_l}]), \quad l = 1, \ldots, L \quad (4.4)$$

where $M_l$ is an $h_{l+1} \times h_l$ deterministic weight matrix multiplied by a diagonal matrix formed by the binary random vector $z_l := [z_{l,1}, z_{l,2}, \ldots, z_{l,h_l}] \in \{0, 1\}^{h_l}$ with entries drawn from a pmf $q_z(z_l; \theta_{z_l})$ parameterized by $\theta_{z_l}$.

If the entries $\{z_{l,i}\}$ are i.i.d. Bernoulli with (identical) probability (w.p.) $\pi$, they effect what is referred to as uniform (across layers and nodes) dropout, which is known to prevent overfitting [118]. Clearly, the parameter set $\theta := \{M_l, \theta_{z_l}\}_{l=1}^{L} = \{M_l\}_{l=1}^{L} \cup \{\pi\}$ fully characterizes $q_\theta(\omega)$. The dropout probability $1 - \pi$ is preselected in practice, while $\{M_l\}_{l=1}^{L}$ can be obtained using the training data by maximizing the log evidence lower bound in (4.2). Nonetheless, integration in (4.2) over all the Bernoulli variables is analytically challenging, while sampling from the Bernoulli pmf is relatively cheap. This prompts approximate yet efficient integration using Monte Carlo estimation. A more detailed account of training Bayesian neural networks can be found in [40, 12, 86]. Moving on, the ensuing subsection deals with detection of adversarial inputs.
4.1.2 Bayesian detection of DNN adversaries

A Bayesian approach to detecting adversarial inputs during the testing phase proceeds by approximating the predictive pdf in (4.1) using the variational surrogate \( q_\theta(\omega) \), as

\[
p(y_\nu|x_\nu, X, Y) \approx \int p(y_\nu|x_\nu, \omega) q_\theta(\omega) d\omega. \tag{4.5}
\]

Deciphering whether a given input \( x_\nu \) is adversarial entails three steps: (S1) parametric modeling of \( q_\theta(\omega) \); (S2) estimating the DNN output uncertainty captured by \( p(y_\nu|x_\nu, X, Y) \); and (S3) declaring \( x_\nu \) as adversarial if the output uncertainty exceeds a certain threshold, and clean otherwise. These steps are elaborated next.

**Step 1:** Parametric modeling of \( q_\theta(\omega) \). Recall that uniform dropout offers a popular special class of \( q_\theta(\omega) \) pdfs, and has been employed in adversary detection [34]. Here, we specify the richer model of \( q_\theta(\omega) \) in (4.3) and (4.4) that will turn out to markedly improve detection performance. Different from uniform dropout, we will allow for (possibly correlated) Bernoulli variables with carefully selected (possibly non-identical) parameters. If such general \( \{\theta_{z_l}\}_{l=1}^L \) can be obtained, matrices \( \{M_l\}_{l=1}^L \) are then found as follows.

Let \( \{W_l^{(TR)}\}_{l=1}^L \) be deterministic weight matrices obtained via non-Bayesian training that we denote as \( \text{TR} \). We will use \( W_l^{(TR)} \) to specify the mean of the random weight matrix \( W_l \) in our Bayesian approach, meaning we choose \( \mathbb{E}_{q_z(z_l;\theta_{z_l})}[W_l]x_{(l-1)} = W_l^{(TR)}x_{(l-1)} \forall l \), where \( x_{(l-1)} \) is the output of the \((l-1)\)st layer for a given input \( x_\nu \) passing through the DNN with deterministic weights \( \{W_l^{(TR)}\}_{l=1}^L \). With \( W_l^{(TR)} \) available, we first design \( q_z(z_l;\theta_{z_l}) \); next, we find \( \theta_{z_l} \); and then \( M_l \), as

\[
M_l = W_l^{(TR)} \text{diag}^\dagger \left( \mathbb{E}_{q_z(z_l;\theta_{z_l})}[z_{l,i}] \right), \quad l = 1, \ldots, L \tag{4.6}
\]

where the pseudo-inverse \( \dagger \) means that inverse entries are replaced with zeros if \( \mathbb{E}_{q_z(z_l;\theta_{z_l})}[z_{l,i}] = 0 \).

**Step 2:** Quantifying the DNN output uncertainty. Since evaluation of \( p(y_\nu|x_\nu, X, Y) \) in (4.5) is prohibitive, one can estimate it using MC sampling. In particular, one can readily obtain MC estimates of (conditional) moments of \( y_\nu \). For instance, its mean and variance can be estimated \(^1\) Such as back propagation based on e.g., a cross-entropy criterion.
as
\[
E_{q_\theta(\omega)}[y_\nu|x_\nu; \{\theta_{zl}\}_{l=1}^L] \simeq \bar{y}_\nu = \frac{1}{R} \sum_{r=1}^R y^{(r)}_\nu
\]
and
\[
\text{Cov}_{q_\theta(\omega)}[y_\nu|x_\nu; \{\theta_{zl}\}_{l=1}^L] \simeq \frac{1}{R} \sum_{r=1}^R y^{(r)}_\nu y^{\top(r)}_\nu - \bar{y}_\nu \bar{y}_\nu^\top
\]
(4.7)

where \(y^{(c)}_\nu\) is the output of the \(r\)-th DNN realized through weights \(\{W^{(r)}_l\}_{l=1}^L\) with input \(x_\nu\). The predictive variance is the trace of Cov \(q_\theta(\omega)[y_\nu|x_\nu; \{\theta_{zl}\}_{l=1}^L]\) that we henceforth abbreviate as Tr(Cov \(q_\theta(\omega)[y_\nu|x_\nu]\)). Given \(x_\nu\), the latter has been used to quantify output uncertainty as \(U(x_\nu) = \text{Tr}(\text{Cov}_{q_\theta(\omega)}[y_\nu|x_\nu])\) [34]. Additional measures of uncertainty will be presented in the next section.

**Step 3: Detecting adversarial inputs.** Given \(U(x_\nu)\), detection of adversarial inputs is cast as testing the hypotheses
\[
\begin{align*}
H_0 : x_\nu &= x^{\text{clean}}_\nu & U(x_\nu) &\leq \tau_0 \\
H_1 : x_\nu &= x^{\text{clean}}_\nu + n^{\text{adv}}_\nu & U(x_\nu) &> \tau_0
\end{align*}
\]
(4.8)

where the null suggests absence of adversarial perturbation (low variance/uncertainty below threshold \(\tau_0\)), while the alternative in effect raises a red flag for presence of adversarial input (high variance/uncertainty above threshold \(\tau_0\)).

We will now proceed to introduce our novel variational distribution model targeting improved detection of adversaries based on uncertainty minimization.

### 4.2 Minimum Uncertainty-based Detection

To design \(q_z(z_i; \theta_{zi})\), we will build on and formalize the sampling scheme in [25] that is employed to specify the joint pmf of the (generally correlated) binary variables \(\{z_{li}\}_{i=1}^{h_l}\) per layer \(l\). To this end, we randomly pick one activation unit output of the \(h_l\) hidden units per layer \(l\); and repeat such a random draw \(C\) times with replacement. Let \(\zeta^{(c)}_l\) denote per draw \(c\) the
$h_l \times 1$ vector variable

$$\zeta_l^{(c)} = [\zeta_{l,1}^{(c)}, \zeta_{l,2}^{(c)} \ldots \zeta_{l,h_l}^{(c)}]^T \sim \text{Categorical}(p_l), \quad c = 1, \ldots, C$$

where each entry $\zeta_{l,i}^{(c)}$ is a binary random variable with

$$\zeta_{l,i}^{(c)} = \begin{cases} 
1 & \text{if draw } c \text{ picks the } i \text{th unit of hidden layer } l \\
0 & \text{otherwise}
\end{cases}$$

and the $h_l \times 1$ vector $p_l$ with nonnegative entries summing up to 1 specifies the Categorical pmf of $\zeta_l^{(c)}$.

With $\|\|$ denoting element-wise binary OR operation on vectors $\{\zeta_l^{(c)}\}_{c=1}^C$, we define next the vector

$$z_l := \zeta_l^{(1)} \| \zeta_l^{(2)} \| \ldots \| \zeta_l^{(C)}.$$ (4.9)

Using $z_l$ as in (4.9) with $\{\theta_{z_l} = p_l\}_{l=1}^L$ to be selected, enables finding the expectation and then $M_l$ in (4.6). Deterministic matrix $M_l$ along with the variates $\{z_l^{(r)}\}_{r=1}^R$ provide the desired DNN realizations to estimate the uncertainty $U(x_\nu; \{p_l\}_{l=1}^L) = \text{Tr}(\text{Cov}_{q_\theta(\omega)}[y_\nu|x_\nu])$ as in (4.7). In turn, this leads to our novel adversarial input detector (cf. (4.8))

\[
\begin{align*}
H_0 & : x_\nu = x_\nu^{\text{clean}} \quad \min_{\{p_l\}_{l=1}^L} U(x_\nu; \{p_l\}_{l=1}^L) \leq \tau_0 \\
H_1 & : x_\nu = x_\nu^{\text{clean}} + n_\nu^{\text{adv}} \quad \text{otherwise}
\end{align*}
\]

(4.10)

where variational parameters $\{p_l\}_{l=1}^L$ are sought such that uncertainty $U(x_\nu; \{p_l\}_{l=1}^L)$ is minimized under $H_0$.

The rationale behind our detector in (4.10) is that given $\tau_0$, minimizing the uncertainty (test statistic) under $H_0$ reduces the probability of false alarms. The probability of detection however, depends on test statistic pdf under $H_1$, in which the adversarial perturbation $n_\nu^{\text{adv}}$ is unknown in practice. The premise here is that due to network instability under $H_1$, the sought probabilities $\{p_l\}_{l=1}^L$ will not reduce uncertainty under $H_1$ as effectively, thus the performance of (4.10) will be better than that of (4.8). To corroborate this, efficient solvers for the proposed minimization task, and extensive tests in lieu of analytical metrics, are in order.
4.2.1 Uncertainty measures

In order to carry the hypothesis test in (4.10), one has options for \( U(x_\nu; \{ p_l \}_{l=1}^L) \) other than the conditional variance. For DNNs designed for classification, mutual information has been recently proposed as a measure of uncertainty [115]

\[
\hat{I}(x_\nu; \{ p_l \}_{l=1}^L) := H(\hat{y}_\nu) - \frac{1}{R} \sum_{r=1}^R H(y_\nu^{(r)})
\]

(4.11)

where superscript \( r \) indexes the pass of input \( x_\nu \) through the \( r \)th DNN realization with corresponding random output \( y_\nu^{(r)} := [y_{\nu,1}^{(r)}, y_{\nu,2}^{(r)}, \ldots, y_{\nu,K}^{(r)}] \) in a \( K \)-class classification task, and \( H(\cdot) \) is the entropy function\(^2\)

\[
H(y_\nu) := - \sum_{k=1}^K y_{\nu,k} \log(y_{\nu,k}).
\]

(4.12)

The test statistic in (4.10) requires finding \( \{ p_l \}_{l=1}^L \) by solving

\[
\min_{\{ p_l \}_{l=1}^L} \hat{I}(x_\nu; \{ p_l \}_{l=1}^L)
\]

(4.13)

which is highly non-convex. However, using Taylor’s expansion of the logarithmic terms in (4.12), one can approximate the mutual information in (4.11) with the variance score \( \text{Tr}(\text{Cov}_q(\omega)[y_\nu]) \) in (4.10), where the conditioning on \( x_\nu \) has been dropped for brevity [115]. As a result, the optimization in (4.13) is approximated as

\[
\min_{\{ p_l \}_{l=1}^L} U(x_\nu; \{ p_l \}_{l=1}^L) = \text{Tr}(\text{Cov}_q(\omega)[y_\nu]).
\]

(4.14)

To solve (4.14), one needs to express the objective in terms of the optimization variables \( \{ p_l \} \) for all layers explicitly. To this end, the following section studies a two-layer network, whose result will then be generalized to deeper models.

\(^2\)Entropy functions in (4.11) are also parameterized by \( \{ p_l \}_{l=1}^L \), but we abbreviate them here as \( H(\hat{y}_\nu) \) and \( H(y_\nu^{(r)}) \).
4.2.2 Simplification of the predictive variance

Aiming at a convenient expression for the cost in (4.14), consider first a two-layer network with input-output (I/O) relationship\footnote{Derivations in this section carry over readily to a more general I/O $y_\nu = \sigma_{\text{softmax}}(W_2 \sigma(W_1 x_\nu) + b_1 + b_2)$ with $b_1$ and $b_2$ deterministic.}

$$y_\nu = \sigma_{\text{softmax}}(W_2 \sigma(W_1 x_\nu))$$

(4.15)

where $W_1, W_2$ are random matrices corresponding to the weights of the two layers as in (4.6), while $\sigma_{\text{softmax}}$ is the softmax memoryless nonlinearity

$$\sigma_{\text{softmax}}(u) := \left[ \begin{array}{c} e^{u_1} \sum_{i=1}^{K} e^{u_i} \\ e^{u_2} \sum_{i=1}^{K} e^{u_i} \\ \vdots \\ e^{u_K} \sum_{i=1}^{K} e^{u_i} \end{array} \right]^\top$$

with $u := [u_1, u_2, \ldots, u_K]^\top$, and the inner $\sigma$ in (4.15) models a general differentiable nonlinearity such as tanh. Although differentiability of the nonlinearities is needed for the derivations in this section, the general idea will be later tested on networks with non-differentiable nonlinearities (such as ReLU) in the experiments.

Given trained weights $\{W_l^{(TR)}\}_{l=1}^2$, and using (4.4) and (4.6), the random weight matrices are found as

$$W_l := M_l \text{diag}(z_l) = W_l^{(TR)} S_l D_l \quad l = 1, 2$$

(4.16)

where $S_l = \text{diag}(z_l)$ denotes the random sampling matrix with pseudo-inverse diagonal mean given by $D_l = \text{diag}^{\dagger}(\mathbb{E}_q(z_l; p_l))[z_l]$. Since $\mathbb{E}[W_l]x_{(l-1)} = W_l^{(TR)}x_{(l-1)}$, the mean of $W_l$ does not depend on $p_l$, while its higher-order moments do.

**Proposition 1.** For the two-layer network in (4.15), the proposed minimization in (4.14) can be approximated by

$$\min_{\{p_l \geq 0, p_l = 1\}_{l=1}^2} \text{Tr}(\text{Cov} W_2 [W_2 \sigma(W_1 x_\nu)]) + \gamma \text{Tr}(\mathbb{E} W_2 [W_2 W_2^\top]) \text{Tr}(\text{Cov} W_1 [W_1 x_\nu])$$

(4.17)

where $\gamma$ is a constant. The solution of (4.17) proceeds in two steps

**Step 1:**

$$p_1^* = \arg \min_{p_1} \text{Tr}(\text{Cov} W_1 [W_1 x_\nu])$$
Step 2: 
\[ p_2^* = \arg \min_{p_2} \text{Tr}(\text{Cov}_{W_2}[W_2\sigma(W_1^{(TR)}x_\nu)]) + \gamma' \text{Tr}(\mathbb{E}_{W_2}[W_2W_2^T]) \]

where \( \gamma' := \gamma' \text{Tr}(\text{Cov}_{W_1}[W_1x_\nu]) \left|_{p_1=p_1^*} \right. \).

Proof. See Appendix B.1.

Remark. The cost in (4.17) approximates that in (4.14) by casting the overall uncertainty minimization as a weighted sum of layer-wise variances. In particular, \( p_1^* \) is the sampling probability vector that minimizes variance score of the first layer. It subsequently influences the regularization scalar \( \gamma' \) in minimizing the second layer variance, which yields the pmf vector \( p_2^* \). This can be inductively generalized to \( L > 2 \) layers. As \( L \) increases however, so do the number of cross terms. For simplicity and scalability, we will further approximate the per-layer minimization by dropping the regularization term, which leads to separable optimization across layers. This is an intuitively pleasing relaxation, because layer-wise variance is minimized under \( \mathcal{H}_0 \), which also minimizes the regularization weight \( \gamma' \).

The resulting non-regularized approximant of step 2 is
\[ p_2^* = \arg \min_{p_2} \text{Tr}(\text{Cov}_{W_2}[W_2\sigma(W_1^{(TR)}x_\nu)]) \]

generalizing to the \( l \)-th layer in an \( L \)-layer DNN as
\[ p_l^* = \arg \min_{p_l} \text{Tr}(\text{Cov}_{W_l}[W_lx_{(l-1)}]) \quad (4.18) \]

where \( x_{(l-1)} \) is the output of the \((l-1)\)st layer, regardless of pmf vectors of other layers \( \{p_{l'}\}_{l' \neq l} \).

4.2.3 Layer-wise variance minimization

Here we will solve the layer-wise variance minimization in (4.18). Using (4.16), the cost can be upper bounded by
\[
\text{Tr}(\text{Cov}_{W_l}[W_l\nu_{(l-1)}]) = \mathbb{E}\left[\|W_l^{(TR)}S_lD_l\nu_{(l-1)} - \mathbb{E}[W_l^{(TR)}S_lD_l\nu_{(l-1)}]\|^2\right] \\
= \mathbb{E}\left[\|W_l^{(TR)}S_lD_l\nu_{(l-1)} - W_l^{(TR)}\nu_{(l-1)}\|^2\right] \\
\leq \|W_l^{(TR)}\|^2 \mathbb{E}\left[\|S_lD_l\nu_{(l-1)} - \nu_{(l-1)}\|^2\right] \\
= \|W_l^{(TR)}\|^2 \sum_{i=1}^{h_l} \mathbb{E}[(S_{li}D_{li}\nu_{(l-1),i} - \nu_{(l-1),i})^2] \\
\]
\[ = \| W_l^{(TR)} \|_2^2 \sum_{i=1}^{h_l} x_{(l-1),i}^2 \mathbb{E}[(S_{ii} D_{ii} - 1)^2] \]
\[ = \| W_l^{(TR)} \|_2^2 \sum_{i=1}^{h_l} x_{(l-1),i}^2 \left( \frac{1}{\pi_{l,i}} - 1 \right) \] (4.19)

where the last equality follows because the \( C \) draws are iid with replacement, and the binary random variables \( z_{l,i} \) reduce to Bernoulli ones with parameter \( \pi_{l,i} = 1 - (1 - p_{l,i})^C \); hence, for \( x_{(l-1),i} \neq 0 \) it holds that \( \mathbb{E}[S_{ii}^2 D_{ii}^2] = 1/\pi_{l,i} \) and \( \mathbb{E}[S_{ii} D_{ii}] = 1 \), which implies that \( \mathbb{E}[(S_{ii} D_{ii} - 1)^2] = (1/\pi_{l,i}) - 2 + 1 \).

Using (4.19), the optimization in (4.18) can be approximately solved by a majorized surrogate as
\[ \min_{p \geq 0, \mathbf{1}^\top p = 1} \sum_{i=1}^{h_l} \frac{1}{1 - (1 - p_{l,i})^C} x_{(l-1),i}^2 \] (4.20)
which is a convex problem that can be solved efficiently as elaborated next.

### 4.3 Solving layer-by-layer minimization

Consider rewriting the layer-wise variance minimization in (4.20) in a general form as
\[ \min_{p \geq 0, \mathbf{1}^\top p = 1} \sum_{i=1}^{h} \alpha_i \left( 1 - (1 - p_i)^C \right) \] (4.21)
where \( \alpha_i := x_{(l-1),i}^2 \) for the \( l \)-th layer. Over the feasible set of the probability simplex, the cost in (4.21) has semi-definite Hessian; thus, it is convex, and can be solved by projected gradient descent iterations. However, \( p \) lies in the probability simplex space of dimension \( h \), the number of hidden nodes in a given layer, and is typically very large. The large number of variables together with possible ill-conditioning can slow down the convergence rate.

To obtain a solver with quadratic convergence rate, we build on the fact that \( h_l \) is usually very large, which implies that \( p_i \ll 1 \) for the practical setting at hand. Using the inequality \( 1 - (1 - p_i)^C \geq 1 - e^{-Cp_i} \), the cost in (4.21) can then be tightly upperbounded, which leads to majorizing (4.21) as
\[ \min_{p \geq 0, \mathbf{1}^\top p = 1} \sum_{i=1}^{h} \frac{\alpha_i}{1 - e^{-Cp_i}} \] (4.22)
The KKT conditions yield the optimal solution of the convex problem in (4.22), as summarized next.

**Proposition 2.** The optimization in (4.22) can be solved with quadratic convergence rate, and the optimum is given by

\[ p^*_i = -\frac{1}{C} \ln \left( \frac{2\rho^* + x^2_{(l-1),i} - \sqrt{[2\rho^* + x^2_{(l-1),i}]^2 - 4\rho^*^2}}{2\rho^*} \right) \]  

(4.23)

where \( \rho^* \) is the solution to the following root-finding problem

\[ \sum_{i=1}^{h} \ln(2\rho + x^2_{(l-1),i}) - \sqrt{[2\rho + x^2_{(l-1),i}]^2 - 4\rho^2} - n \ln(2\rho) + C = 0. \]

**Proof.** See Appendix B.2

### 4.3.1 Approximate variance minimization for small \( C \)

For small values of \( C \), it holds that \((1 - p_i)^C > 1 - C p_i\); hence, the Bernoulli parameter \( \pi_i = 1 - (1 - p_i)^C \) can be approximated by its upperbound \( C p_i > \pi_i \). With this we can approximate the cost in (4.20), as

\[ \min_{p \geq 0, \mathbf{1}^T p = 1} \sum_{i=1}^{h} \frac{\alpha_i}{C p_i}. \]  

(4.24)

Using the Lagrangian and the KKT conditions, we then find \( p^*_i = \sqrt{\alpha_i / \sum_{j=1}^{h} \sqrt{\alpha_j}} \), which for the \( l \)-th layer is expressible as

\[ p^*_{(l-1),i} = \frac{|x_{(l-1),i}|}{\sum_{j=1}^{h_l} |x_{(l-1),j}|}. \]  

(4.25)

This approximation provides analytical justification for the heuristic approach in [25], where it is proposed to sample with probabilities proportional to the magnitude of the hidden unit outputs. However, there remains a subtle difference, which will be clarified in Section 4.5.

Approximating (4.22) with (4.24) can be loose for large values of \( C \), which motivates our next approximation.
4.3.2 Approximate variance minimization for large $C$

Building on the tight approximation in (4.22), one can further approximate the variance for large $C$ as

$$\min_{p \geq 0, \sum p = 1} \sum_{i=1}^{h} \frac{\alpha_i}{1 - e^{-Cp_i}} \simeq \min_{p \geq 0, \sum p = 1} \sum_{i=1}^{h} \alpha_i (1 + e^{-Cp_i})$$

where we have used $(1 - \delta)^{-1} \simeq 1 + \delta$ as a tight approximation for $0 < \delta \ll 1$. This leads to the minimization

$$\min_{p \geq 0, \sum p = 1} \sum_{i=1}^{n} \alpha_i e^{-Cp_i}$$

which again is a convex problem, whose solution can be obtained using the KKT conditions that lead to

$$-C\alpha_i e^{-C\hat{p}_i^*} + \lambda = 0 \quad \forall i$$

where $\lambda$ is the Lagrange multiplier. Under the simplex constraint on the $\{p_i\}$, this leads to the optimal

$$\hat{p}_i^* = \left[ \frac{1}{C} \ln \left[ C \alpha_i^2 \{l-1,i\} + \hat{\beta}^* \right] + \hat{\beta}^* \right]_+$$

(4.26)

with $[.]_+$ denoting the projection on the positive orthant, and the normalization constant $\beta := -\ln \lambda/C$ having optimal value

$$\hat{\beta}^* = \frac{1 - \sum_{i=1}^{h} \hat{p}_i^* \{\hat{p}_i^* > 0\}}{\sum_{i=1}^{h} \{\hat{p}_i^* > 0\}}.$$

Although the solution to the fixed point condition cannot be obtained at one shot, and may require a few iterations to converge, in practice we only perform it once and settle with the obtained approximate solution $\{\hat{p}_i^*\}_{i=1}^{h}$.

4.4 Practical issues

The present section deals with efficient implementation of the proposed approach in practice, and establishes links with state-of-the-art Bayesian detection methods.
4.4.1 Efficient implementation via non-uniform dropout

The proposed defense builds on modeling the variational pdf \( q_\theta(\omega) \) using a sampling-with-replacement process. Performing the proposed process however, may incur overhead complexity during inference when compared to the inexpensive dropout alternative outlined in Sec. 4.1.1. To reduce this complexity, one can implement our approach using efficient approximations, while leveraging the sampling probabilities learned through our uncertainty minimization.

Reflecting on the binary variables \( \{ z_{l,i} \} \) that model the pickup of the hidden node \( i \) in the overall sampling process in (4.9), one can approximate the joint pmf of \( \{ z_{l,i} \}_{i=1}^{h_l} \) as

\[
q_z(z_l; p_l) \simeq h_l \prod_{i=1}^{h_l} q(z_{l,i}; p_{l,i})
\]

where random variables \( \{ z_{l,i} \}_{i} \) are now viewed as approximately independent non-identical Bernoulli variables with parameters \( \{ \pi_{l,i} \}_{i=1}^{h_l} \); that is, \( z_{l,i} \sim \text{Bernoulli}(\pi_{l,i}) \) for \( i = 1, \ldots, h_l \), where \( \pi_{l,i} = 1 - (1 - p_{l,i})^C \).

Although (4.27) is an approximation, it provides insight but also an efficient implementation of the sampling process. In fact, the proposed optimization in (4.21) can now be viewed as an optimization over the non-uniform dropout probabilities, coupled implicitly through the hyper-parameter \( C \), whose selection guarantees a certain level of randomness. This is to be contrasted with finding optimal dropout probabilities - a task requiring grid search over an \( h_l \)-dimensional space for layer \( l \), where \( h_l \) can be hundreds of thousands to millions in CNNs classifying high-quality images. Interestingly, the proposed convex optimization simplifies the high-dimensional grid-search into a scalar root-finding task, whose solution can be efficiently found with super-linear (quadratic) convergence rate.

4.4.2 Placement and adjustment of the sampling units

It has been argued that CNN layers at different depths can provide extracted features with variable levels of expressiveness [41]. On a par with this, one can envision the defense potential at different depths by incorporating sampling units across say \( B \) blocks of the network as listed in Tables 4.2 and 4.3. In particular, the dropout defense has been mostly utilized at the last layer after flattening [115], whereas here we consider the potential of sampling at earlier layers that has gone mostly under-explored so far. This can in turn result in Bayesian DNN-based classifiers
Bayesian approaches to detecting adversaries in DNNs

Uniform dropout  Minimum uncertainty (variance) based
(Dropout)    (non-uniform sampling)

Fixed/Deterministic     Dynamic
Exact     Linear approx.   Logarithmic approx.    Linear approx.   Logarithmic approx.
(VM-exact) (VM-lin) (VM-log) (SAP or DVM-lin) (DVM-log)

Figure 4.1: Overview of Bayesian adversary detection schemes

with robustness to adversarial attacks, as optimal sampling at the initial layers maybe crucial for correct detection of the adversarial input. We henceforth refer to a DNN (or CNN) equipped with random sampling as the detection network, and the original one without the sampling units as the full network.

Similar to the pick up probability $\pi$ in uniform dropouts, the number of draws $C$ in our approach is a hyper parameter that controls the level of randomness present in the detection network. Qualitatively speaking, the smaller number of units (smaller $C$) is picked per layer, the larger ‘amount of randomness’ emerges (further $\pi_{l,i}$ is from 1). This can lead to forward propagating not as informative (under-sampled) features, meaning not representative of the clean image, and can thus cause unreliable detection. A large $C$ on the other hand, increases the probability to pick up units per layer, which requires a large number of MC realizations for reliable detection, otherwise small randomness will lead to miss-detection. At the extreme, very large $C$ renders the detection and full networks identical, thus leading to unsuccessful detection of adversarial inputs. In a nutshell, there is a trade-off in selecting $C$, potentially different for the initial, middle, and final layers of a given CNN.

Fig. 4.1 categorizes existing and the novel randomization-based approaches to detecting adversarial inputs.

**Uniform dropout.** In this method, units are independently dropped w.p. $1 - \pi$, and sampled (picked) w.p. $\pi \forall l, i$.

**Non-uniform dropout using variance minimization.** Dropout here follows the scheme in subsection 4.4.1 for which we pursue the following two general cases with deterministic and
Algorithm 9 Adversary detection - fixed \( \{p_{l,i}\} \)

**Input:** Test image \( x_\nu, B, C, R \) and \( \tau_0 \)

Pass image \( x_\nu \) through full network; find \( \{x_{(l-1),i}\} \)

Use \( \{x_{(l-1),i}\} \) to obtain \( \{p_{l,i}\} \) via (4.23), (4.25) or (4.26)

for \( r = 1, 2, \ldots, R \) do
  Collect output class \( y_\nu^{(r)} \)
  Estimate the mutual information (MI) of \( \{y_\nu^{(r)}\}_{r=1}^R \)
end for

**Output:** Declare adversary if MI exceeds threshold \( \tau_0 \)

*dynamic* probabilities.

**(C1) Variance minimization with fixed probabilities.** In this case, the image is first passed through the full network to obtain the values \( \{x_{(l-1),i}\} \) of the unit outputs per hidden layer. These are needed to determine the non-uniform dropout probabilities \( 1 - p_{l,i} \) (thus \( \pi_{l,i} \) and then the index of the units to sample) via *exact*, *linear*, or *logarithmic* approximations given in (4.23), (4.25) and (4.26), respectively, referred to as VM-exact, VM-lin, and VM-log; see Fig. 4.2-a.

Despite parallel MC passes in the proposed class of sampling with fixed probabilities (step 3 in Fig. 4.2-a), the first step still imposes a *serial* overhead in detection since the wanted probabilities must be obtained using a pass through the full network. Our approach to circumventing this overhead is through approximation using the following class of sampling with *dynamic* probabilities.

**(C2) Variance minimization with dynamic probabilities.** Rather than finding the sampling probabilities beforehand, \( p_{l,i}^{(r)} \) are determined *on-the-fly* as the image is passed through the detection network with the units sampled per layer. As a result, the observed unit values are random (after passing through at least one unit sampled), and are different across realizations. In order to mitigate solving many optimization problems, variance minimization with dynamic probabilities is only implemented via *linear* and *logarithmic* approximations (4.25) and (4.26); and are referred to as DVM-lin and DVM-log, respectively; see Fig. 4.2-b.

It is interesting to note that DVM-lin corresponds to the proposed stochastic activation
Algorithm 10 Adversary detection - dynamic $\{p_{i,t}\}$

**Input:** Test image $x_\nu$, $B$, $C$, $R$ and $\tau_0$

**for** $r = 1, 2, \ldots, R$ **do**

Collect $y_\nu^{(r)}$ after passing $x_\nu$ through the detection network with units picked with dynamic probabilities obtained (exactly or approximately) using the observed values

Estimate the mutual information (MI) of $\{y_\nu^{(r)}\}_{r=1}^R$

**end for**

**Output:** Declare adversary if MI exceeds threshold $\tau_0$
pruning (SAP) in [25], with

\[
\mathbf{p}_{l,r}^{\text{SAP}} = \left[ \frac{|x_{(l-1),1}^{(r)}|}{\sum_{i=1}^{h_l} |x_{(l-1),i}^{(r)}|}, \frac{|x_{(l-1),2}^{(r)}|}{\sum_{i=1}^{h_l} |x_{(l-1),i}^{(r)}|}, \ldots, \frac{|x_{(l-1),h_l}^{(r)}|}{\sum_{i=1}^{h_l} |x_{(l-1),i}^{(r)}|} \right]
\]

where \(x_{(l-1),i}^{(r)}\) is the output of the \(i\)-th activation unit of the \(l\)-th layer in the \(r\)-th realization for input \(x\).

Figure 4.1 provides an overview of the sampling methods, while Algorithms 9 and (10) outline the two proposed variance minimization-based detection methods in pseudocode.

### 4.5 Numerical tests

**Algorithm 11** Layer-wise minimum variance solver

**Solve:** \( \min_{\mathbf{p} \geq 0, \mathbf{1}^T \mathbf{p} = 1} \sum_{i=1}^{h} \frac{\alpha_i}{1 - (1 - p_i)C} \)

**Input:** \([\alpha_1, \alpha_2, \ldots, \alpha_h], C\)

Using bisection and initialization \(\rho_0 = \sum_{i=1}^{h} \alpha_i / h\), find the root \(\rho^*\) for

\[
\sum_{i} \ln(2\rho + \alpha_i - \sqrt{(2\rho + \alpha_i)^2 - 4\rho^2}) - n \ln(2\rho) + C = 0
\]

Set \(p_i^* = -\frac{1}{C} \ln \left( \frac{2\rho^* + \alpha_i - \sqrt{(2\rho^* + \alpha_i)^2 - 4\rho^*^2}}{2\rho^*} \right) \forall i\)

Set \(\pi_i^* = 1 - (1 - p_i^*)^C \forall i\)

**Output:** Nonuniform dropout pmf \(\pi = [\pi_1 \ldots \pi_h]^T\)

In this section, we test the effectiveness of the proposed Bayesian sampling method for detecting various adversarial attacks on CNNs used for image classification. In order to address the raised issue in [17], classification of the CIFAR10 image dataset using ResNet20 as well as the high-resolution cats-and-dogs images using ResNet34 networks [54] are tested. A short summary of the two networks and datasets can be found in Tables 4.1, 4.2 and 4.3. In order to investigate the issue around placement of the sampling units, we will place them after ReLU activation layers in different “blocks” \((B)\) of the ResNet20 and ResNet34 networks, as listed in
<table>
<thead>
<tr>
<th>Dataset</th>
<th>image size</th>
<th># train</th>
<th># val.</th>
<th># test</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIFAR10</td>
<td>32 x 32</td>
<td>50,000</td>
<td>2,000</td>
<td>8,000</td>
</tr>
<tr>
<td>Cats-and-dogs</td>
<td>224 x 224</td>
<td>10,000</td>
<td>2,000</td>
<td>13,000</td>
</tr>
</tbody>
</table>

Table 4.1: CIFAR10 and cats-and-dogs image-classification datasets

<table>
<thead>
<tr>
<th>name</th>
<th>output-size</th>
<th>20 layers</th>
<th>#sampling units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block1</td>
<td>32 x 32</td>
<td>[ 3 x 3, 16]</td>
<td>1</td>
</tr>
<tr>
<td>Block2</td>
<td>32 x 32</td>
<td>3 x 3, 16 x 3</td>
<td>6</td>
</tr>
<tr>
<td>Block3</td>
<td>16 x 16</td>
<td>3 x 3, 32 x 3</td>
<td>6</td>
</tr>
<tr>
<td>Block4</td>
<td>8 x 8</td>
<td>3 x 3, 64 x 3</td>
<td>6</td>
</tr>
<tr>
<td>Block5</td>
<td>1 x 1</td>
<td>average pool, 64-d fully conn., softmax</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4.2: ResNet20 architecture on CIFAR10 dataset

Tables 4.2 and 4.3 Numerical tests are made available online.

4.5.1 CIFAR10 dataset

ResNet20 is trained using 20 epochs with minibatches of size 128. Adversarial inputs are crafted on the corresponding MC network as in [115], using the fast gradient sign method (FGSM) [43], the basic iterative method (BIM) [67], the momentum iterative method (MIM) [26], and the Carlini-and-Wagner (C&W) [18] attacks. Parameters of the attacks as well as test accuracy of the MC network on clean and adversarial inputs are reported in Tables 4.5 and 4.4.

Placement parameter $B$ and sampling parameters $C$ for variance minimization methods as well as the dropout probability for uniform dropout are selected by cross validation. To clarify the suboptimality gap between the exact and approximate variance minimization with deterministic sampling probabilities, we have cross-validated the parameters for VM-exact, and reused them for VM-lin and VM-log approximates.

https://github.com/FatemehSheikholeslami/variance-minimization
Table 4.3: ResNet34 architecture on cats-and-dogs dataset

<table>
<thead>
<tr>
<th>Block</th>
<th>output-size</th>
<th>34 layers</th>
<th>#sampling units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block 1</td>
<td>112 x 112</td>
<td>[7 x 7, 64], 3x3 max-pool</td>
<td>2</td>
</tr>
<tr>
<td>Block 2</td>
<td>56 x 56</td>
<td>[3 x 3, 64], [3 x 3, 64] x 3</td>
<td>6</td>
</tr>
<tr>
<td>Block 3</td>
<td>28 x 28</td>
<td>[3 x 3, 128], [3 x 3, 128] x 4</td>
<td>8</td>
</tr>
<tr>
<td>Block 4</td>
<td>14 x 14</td>
<td>[3 x 3, 256], [3 x 3, 256] x 6</td>
<td>12</td>
</tr>
<tr>
<td>Block 5</td>
<td>7 x 7</td>
<td>[3 x 3, 512], [3 x 3, 512] x 3</td>
<td>6</td>
</tr>
<tr>
<td>Block 6</td>
<td>1 x 1</td>
<td>average pool, 1000-d fc, softmax</td>
<td>1</td>
</tr>
</tbody>
</table>

The sampling parameter is selected as \( C = f \times \text{nnz}(x_l) \) for the \( l \)-th layer sampling unit, where \( \text{nnz}(.) \) denotes the number of non-zero entries\(^5\) and \( f \) is the sampling ratio varied in \( f \in \{0.6, 0.7, 0.8, 0.9, 1.0, 1.5, 2.0, 3.0, 4.0\} \). Probability in uniform dropout is also varied as \( \pi_{\text{drop}} \in \{0.1, 0.2, \ldots, 0.7\} \), and the number of MC runs is \( R = 20 \).

In order to properly evaluate accuracy in detection of adversarial images, we only aim at detecting the test samples that are correctly classified by the full network, and misclassified after the adversarial perturbation. The detection performance is then reported in terms of the receiver operating characteristic (ROC) curve in Fig. 4.3 obtained by varying the threshold parameter \( \tau_0 \). The exact area-under-curve values along with parameters \( B, f, \pi_{\text{drop}} \) are also reported in Tables 4.6, 4.7, 4.8, and 4.9 highlighting the improved detection via the proposed variance minimization approach.

Furthermore, in order to target more realistic scenarios, where attack generation is unknown and may indeed be crafted via various methods, we have also tested the performance against a “combination attack,” in which the adversarial input crafted with all 7 settings of attacks are considered. This indeed corroborates that placement of the sampling units in the fourth block

---

\(^5\)This selection is chosen by taking into account the fact that, only non-zero samples will be dropped upon not being selected, while zero entries will remain unchanged regardless of the sampling outcome.

\(^6\)Since the sampling procedure is modeled with replacement, fraction \( f \) may be selected greater than 100%.
Figure 4.3: ROC-curve of different attack-detection sampling schemes on CIFAR10 dataset.
Table 4.4: Attack parameters and test accuracy on clean, FGSM and BIM adversarial input in CIFAR10 dataset.

<table>
<thead>
<tr>
<th>Attack parameters</th>
<th>clean</th>
<th>FGSM</th>
<th>BIM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>–</td>
<td>–</td>
<td>norm: $\infty$ # iter: 20 $\epsilon_{iter}: 0.005$</td>
</tr>
<tr>
<td>$\epsilon = 10/255$</td>
<td>64.87%</td>
<td>56.91%</td>
<td>5.2%</td>
</tr>
<tr>
<td>$\epsilon = 20/255$</td>
<td>5.0%</td>
<td>5.0%</td>
<td></td>
</tr>
</tbody>
</table>

Class. Acc. 91.5% 64.87% 56.91% 5.2% 5.0%

Table 4.5: Attack parameters and test accuracy on BIM and C&W adversarial input in CIFAR10.

<table>
<thead>
<tr>
<th>Attack parameters</th>
<th>MIM</th>
<th>C&amp;W</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>norm: $\infty$ # binary search: 10 $\epsilon_{iter}: 10$</td>
<td></td>
</tr>
<tr>
<td>$\epsilon = 10/255$</td>
<td>5.4%</td>
<td>5.1%</td>
</tr>
<tr>
<td>$\epsilon = 20/255$</td>
<td>11.7%</td>
<td></td>
</tr>
</tbody>
</table>

Class. Acc. 5.4% 5.1% 11.7%

along with careful tuning of the sampling probabilities via VM-exact provides the highest curve against combination of attacks, while its approximations follow in performance, outperforming uniform dropout. For further discussion on sensitivity against parameter selection, see Appendix B.3.

Table 4.6: AUC-ROC of different attack-detection sampling schemes on CIFAR10 test set against FGSM and MIM attacks. Higher values indicate better detection.

<table>
<thead>
<tr>
<th>Sampling Method</th>
<th>FGSM Attack $\epsilon = 10$</th>
<th>FGSM Attack $\epsilon = 20$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Parameters</td>
<td>AUC</td>
</tr>
<tr>
<td>VM</td>
<td>81.9</td>
<td></td>
</tr>
<tr>
<td>VM-log</td>
<td>$(B, f) = (4, 2.0)$</td>
<td>79.3</td>
</tr>
<tr>
<td>VM-linear</td>
<td>77.9</td>
<td></td>
</tr>
<tr>
<td>DVM-log</td>
<td>$(B, f) = (4, 3.0)$</td>
<td>78.4</td>
</tr>
<tr>
<td>SAP</td>
<td>$(B, f) = (2, 4.0)$</td>
<td>79.3</td>
</tr>
<tr>
<td>Dropout</td>
<td>$(B, \pi_{dp}) = (5, 0.1)$</td>
<td>77.0</td>
</tr>
</tbody>
</table>
a) FGSM attack with $\epsilon = 10$

b) MIM attack with $\epsilon = 10$

c) BIM attack with $\epsilon = 10$

d) FGSM attack with $\epsilon = 20$

e) MIM attack with $\epsilon = 20$

f) BIM attack with $\epsilon = 20$

g) C&W attack

h) Combination attack

Figure 4.4: ROC-curve of different attack-detection sampling schemes on cats-and-dogs dataset.
Table 4.7: AUC-ROC of different attack-detection sampling schemes on CIFAR10 test set against FGSM and MIM attacks. Higher values indicate better detection.

<table>
<thead>
<tr>
<th>Sampling Method</th>
<th>MIM Attack</th>
<th>( \epsilon = 10 )</th>
<th>( \epsilon = 20 )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Parameters</td>
<td>AUC</td>
<td>Parameters</td>
</tr>
<tr>
<td>VM</td>
<td>(4, 4.0)</td>
<td>74.4</td>
<td>(4, 4.0)</td>
</tr>
<tr>
<td>VM-log</td>
<td>(B, f = (4, 4.0))</td>
<td>71.4</td>
<td>(B, f = (4, 4.0))</td>
</tr>
<tr>
<td>VM-linear</td>
<td>(4, 4.0)</td>
<td>71.8</td>
<td>(4, 4.0)</td>
</tr>
<tr>
<td>DVM-log</td>
<td>(5, 4.0)</td>
<td>70.3</td>
<td>(4, 4.0)</td>
</tr>
<tr>
<td>SAP</td>
<td>(2, 3.0)</td>
<td>73.8</td>
<td>(3, 4.0)</td>
</tr>
<tr>
<td>Dropout</td>
<td>(5, 0.1)</td>
<td>69.6</td>
<td>(5, 0.2)</td>
</tr>
</tbody>
</table>

Table 4.8: AUC-ROC of different attack-detection sampling schemes on CIFAR10 test set against FGSM, C&W, and combination attacks. Higher values indicate better detection.

<table>
<thead>
<tr>
<th>Sampling Method</th>
<th>BIM Attack</th>
<th>( \epsilon = 10 )</th>
<th>( \epsilon = 20 )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Parameters</td>
<td>AUC</td>
<td>Parameters</td>
</tr>
<tr>
<td>VM</td>
<td>(4, 4.0)</td>
<td>67.0</td>
<td>(4, 4.0)</td>
</tr>
<tr>
<td>VM-log</td>
<td>(B, f = (4, 4.0))</td>
<td>62.6</td>
<td>(B, f = (4, 4.0))</td>
</tr>
<tr>
<td>VM-linear</td>
<td>(4, 4.0)</td>
<td>63.1</td>
<td>(4, 4.0)</td>
</tr>
<tr>
<td>DVM-log</td>
<td>(1, 3.0)</td>
<td>64.8</td>
<td>(1, 4.0)</td>
</tr>
<tr>
<td>SAP</td>
<td>(2, 1.5)</td>
<td>71.2</td>
<td>(2, 1.5)</td>
</tr>
<tr>
<td>Dropout</td>
<td>(2, 0.1)</td>
<td>69.6</td>
<td>(2, 0.1)</td>
</tr>
</tbody>
</table>

Table 4.9: AUC-ROC of different attack-detection sampling schemes on CIFAR10 test set against FGSM, C&W, and combination attacks. Higher values indicate better detection.

<table>
<thead>
<tr>
<th>Sampling Method</th>
<th>C&amp;W Attack</th>
<th>Combination Attack</th>
<th>AUC</th>
<th>AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Parameters</td>
<td>AUC</td>
<td>AUC</td>
<td></td>
</tr>
<tr>
<td>VM</td>
<td>(4, 3.0)</td>
<td>81.6</td>
<td>76.0</td>
<td></td>
</tr>
<tr>
<td>VM-log</td>
<td>(B, f = (4, 4.0))</td>
<td>79.6</td>
<td>(B, f = (4, 4.0))</td>
<td>72.8</td>
</tr>
<tr>
<td>VM-linear</td>
<td>(4, 4.0)</td>
<td>79.3</td>
<td>72.9</td>
<td></td>
</tr>
<tr>
<td>DVM-log</td>
<td>(5, 0.8)</td>
<td>77.8</td>
<td>71.4</td>
<td></td>
</tr>
<tr>
<td>SAP</td>
<td>(5, 4.0)</td>
<td>79.7</td>
<td>74.2</td>
<td></td>
</tr>
<tr>
<td>Dropout</td>
<td>(5, 0.2)</td>
<td>78.8</td>
<td>71.7</td>
<td></td>
</tr>
</tbody>
</table>
### 4.5.2 Cats-and-dogs dataset

Tests are also carried out for the cats-and-dogs dataset\(^7\) which consists of high-quality images classified into binary classes of cats and dogs. Images are resized to 224 × 224, and are classified

---

Table 4.13: AUC-ROC of different attack-detection sampling schemes on cats-and-dogs test set against BIM attack. Higher values indicate better detection.

<table>
<thead>
<tr>
<th>Sampling Method</th>
<th>C&amp;W Attack</th>
<th>BIM Attack</th>
<th>Combination Attack</th>
</tr>
</thead>
<tbody>
<tr>
<td>VM</td>
<td>(B, f) = (1, 3.0)</td>
<td>71.6</td>
<td>70.4</td>
</tr>
<tr>
<td>VM-log</td>
<td>(B, f) = (2, 0.8)</td>
<td>56.9</td>
<td>(B, f) = (2, 0.7)</td>
</tr>
<tr>
<td>VM-linear</td>
<td>71.9</td>
<td></td>
<td>68.2</td>
</tr>
<tr>
<td>DVM-log</td>
<td>(B, f) = (4, 3.0)</td>
<td>69.6</td>
<td>(B, f) = (4, 3.0)</td>
</tr>
<tr>
<td>SAP</td>
<td>(B, f) = (1, 3.0)</td>
<td>63.8</td>
<td>(B, f) = (6, 1.0)</td>
</tr>
<tr>
<td>Dropout</td>
<td>(B, π_{dropout}) = (5, 0.6)</td>
<td>63.1</td>
<td>(B, π_{dropout}) = (6, 0.1)</td>
</tr>
</tbody>
</table>

Table 4.14: AUC-ROC of different attack-detection sampling schemes on cats-and-dogs test set against C&W and combination attacks. Higher values indicate better detection.

<table>
<thead>
<tr>
<th>Sampling Method</th>
<th>C&amp;W Attack</th>
<th>BIM Attack</th>
<th>Combination Attack</th>
</tr>
</thead>
<tbody>
<tr>
<td>VM</td>
<td>(B, f) = (1, 3.0)</td>
<td>71.4</td>
<td>70.7</td>
</tr>
<tr>
<td>VM-log</td>
<td>(B, f) = (2, 0.8)</td>
<td>70.3</td>
<td>(B, f) = (1, 3.0)</td>
</tr>
<tr>
<td>VM-linear</td>
<td>66.7</td>
<td></td>
<td>69.4</td>
</tr>
<tr>
<td>DVM-log</td>
<td>(B, f) = (4, 3.0)</td>
<td>63.9</td>
<td>(B, f) = (4, 3.0)</td>
</tr>
<tr>
<td>SAP</td>
<td>(B, f) = (1, 3.0)</td>
<td>67.1</td>
<td>(B, f) = (1, 3.0)</td>
</tr>
<tr>
<td>Dropout</td>
<td>(B, π_{dropout}) = (5, 0.6)</td>
<td>60.1</td>
<td>(B, π_{dropout}) = (6, 0.1)</td>
</tr>
</tbody>
</table>

using ResNet34 [54]. Weights of the convolutional layers are transferred from the network trained on the ImageNet dataset. This is subsequently followed by a dropout, 1000 × 2 fully-connected and softmax layer, whose weights are trained using 10,000 images; see Table 4.1. The FGMS, BIM, MIM, an C&W attacks are crafted, and parameters are reported in Table 4.10. Detection parameters are similarly selected by using the validation set and varying B ∈ \{1, 2, 3, 4, 5, 6\}, f ∈ \{0.6, 0.7, 0.8, 0.9, 1.0, 1.5, 2.0, 3.0\}, where C = f × \text{nnz}(x_l), π_{dropout} ∈ \{0.1, 0.2, \ldots, 0.7\}, and the number of MC runs is R = 20.

Fig. 4.4 plots the ROC curve for detection of adversarial versus clean images, and defense parameters are reported in Tables 4.11, 4.12, 4.13, and 4.14 quantifying the accuracy of attack detection across different methods. As with CIFAR10, tests are also extended to a combination attack, where detection is performed against the combination of all seven attacks with a fixed set

---

8https://github.com/qubvel/classification_models
of defense parameters $B$, $f$, and $\pi_{\text{drop}}$.

Interestingly, it is observed that for small values of $f$, the linear approximation for variance minimization (VM-lin) follows the performance of the exact variance minimization (VM-exact) closely for FGSM, MIM, and C&W attacks, whereas the logarithmic approximation (VM-log) exhibits a large gap in performance. In contrast, for large values of $f$, VM-log demonstrates a smaller optimality gap with VM-exact as opposed to VM-lin; see Figs. 4.4(c) and (f). This corroborates our approximations in (4.24) and (4.26), providing high-performance low-complexity substitutes for the exact variance-minimization solver in both small and large sampling regimes, that is $f < 1$ and $f > 1$. Similarly, improved performance of the logarithmic approximates versus the linear ones are also corroborated in the high-quality cats-and-dogs images versus CIFAR10, due to higher $C$ resulting from higher dimensional vectors $x_{(l)}$ in the hidden layers.

The ROC curves further demonstrate that performance of the deterministic sampling probabilities, obtained by passing the image through the full network, are often superior to the dynamic ones (SAP and DVM-log), among which DVM-log demonstrates better performance.
Chapter 5

Summary and Future Directions

This thesis contributes to the design of theoretically-grounded, robust, and low-complexity algorithms for adversarial, online, and large-scale machine learning settings. In this final chapter, the main results are summarized, and some possible directions for future research are pointed out.

5.1 Thesis summary

Targeting kernel-based learning machines that are affordable over large-scale data streams under limited memory and computational resources, Chapter 2 aims at devising scalable algorithms for nonlinear feature extraction, applicable in a multitude of tasks such as KPCA for clustering, as well as KSVM for classification and regression. To this end, we have formulated a double-blind optimization problem for dynamic subspace tracking and developed efficient solvers for which online and budgeted modifications are also proposed. These can accommodate sequential processing and adhere to budgeted memory and stringent computational constraints. Furthermore, the performance gap due to the budget constraints relative to their ‘unbudgeted’ counterparts has been derived, and efficiency of the proposed method has been demonstrated through numerical tests on time-varying datasets.

Considering the vital importance of networks as the workhorse in modeling complex real-world interactions, identification of overlapping communities is studied in Chapter 3. Upon recognizing that a network is in fact the union of its egonets, a novel network representation using multi-way data structures is advocated in this contribution. To leverage the rich structure
offered through multi-dimensional tensors, a constrained tensor approximation, titled EgoTen, is introduced using PARAFAC decomposition, and is handled via alternating minimization. The obtained factors provide soft community memberships, and are utilized for potential multi-community assignment of nodes, leading to the successful detection of overlapping communities. The scope of the novel algorithm is further broadened for application to large-scale real-world networks, such as YouTube and Facebook platforms, in which a top-down community identification via successive application of the EgoTen scheme is advocated. Test results demonstrate that the rich structure of tensors indeed leads to a remarkable improvement in detection performance across multiple datasets.

Finally, Chapter 4 deals with the presence of adversaries for image classification in the context of safety-critical environments such as self-driving cars and robotics. Aiming at successful detection of adversarial inputs, a Bayesian approach is pursued, where a general class of variational inference approximations is advocated through our proposed sampling procedure, utilized across different depths in the network. This provides a means of estimating network uncertainty for a given input. Building on the premise that adversarial inputs are non-compliant with the natural-data manifold, thus inducing higher network uncertainty than clean images, one can declare such input as adversarial during the test phase. The proposed sampling units are modular, and provide tunable probabilities which are optimized under the proposed objective. Exact and low-complexity approximate solvers are investigated, and connections with state-of-the-art detection schemes is provided. Finally, the chapter is concluded by extensive tests on various datasets and attack schemes over state-of-the-art CNNs, demonstrating remarkable improvement in detection accuracy, thus corroborating the importance and effectiveness of the proposed adjustable sampling units.

5.2 Future research

The promising results in this thesis open up interesting directions for a number of future research topics. The following subsections discuss a few such directions.

5.2.1 Kernel-based representation of networks

Building on the nonlinear feature extraction schemes developed in Chapter 2, one can utilize the novel idea for measurements collected over entities/nodes in a network, such as brain voxels or
Facebook users. Measurements over nodal activities across time give rise to 2-D data-features, describing the temporal behavior of the network. In such scenarios, the task of data analysis is often equipped with known characterizations of the network such as the adjacency matrix describing the biological, functional, geographical, or social connectivity patterns among the entities. Thus, when applying nonlinear feature extraction over such 2-D measurements, one is highly motivated to incorporate this information in order to obtain more informative features. To this end, one can develop frameworks for efficient and online extraction of nonlinear features of 2-D measurements over networks while utilizing the known network topology. Sample-complexity analysis for the batch solver and regret bound analysis for the online framework are also of interest along this direction.

5.2.2 Joint utilization of egonet-tensors and extra nodal features

The proposed tensor-based community detection is of prime interest especially in networks where due to privacy or limited information, no extra nodal features are available [137]. In these cases, extraction of egonet connections serves as a systematic means of “feature” extraction on the network nodes. However, in certain applications extra nodal features are also available. For instance, in the co-authorship network extra information such as affiliated university, education, and title of the authors are often known, or in social media networks user features such as gender, city of residence, education and other features are sometimes displayed to the public. Thus, utilization of such features can intuitively facilitate the analysis of complex networks.

The emergent research direction is to incorporate such extra nodal features in the proposed PARAFAC decomposition, so that the community affiliations offered by the egonet-tensor decomposition jointly minimize the factorization objective, as well as certain distance measures over the available nodal features. Our initial proposition is to consider the minimization

$$\arg \min_{A, B, C; \{\bar{f}_k\}_{k=1}^K} \left\{ \| W - \sum_{k=1}^K a_k \circ b_k \circ c_k \|_F^2 + \lambda (\| A \|_F^2 + \| B \|_F^2) \right\} + \gamma \sum_{n=1}^N \| f_n - \sum_{k=1}^K c_{nk} \bar{f}_k \|_2^2$$

s.t. $A \geq 0, B \geq 0, C \geq 0, \sum_{k=1}^K c_{nk} = 1 \quad \forall n = 1, 2, ..., N$

where $f_n$ is the feature vector of node $n$, and $\bar{f}_k$ is the feature-centroid associated with community $k$, whose distance is minimized with respect to its community members via the $\ell_2$-norm.
regularization. Potential benefits of joint consideration of egonet tensor decomposition and nodal features is of interest, and alternative methods for incorporating nodal features will be investigated.

5.2.3 Multi-dimensional egonet-tensors for intruder detection

Among the main advantages of the proposed egonet tensor for analysis is the fact that it provides an enhanced representation of the network. Thus, the novel approach can be further exploited in targeting other analytic tasks such as anomaly/intruder detection \cite{7}. Low-rank matrix factorization has been widely used in the detection of anomalous data, whose noncompliance with the inherent low-rank model has been captured through the $\ell_1$-norm regularization \cite{81}. Similar ideas can be exploited in the low-rank tensor decomposition of (3.1). One proposed technique is to modify the decomposition as

$$\arg \min_{A, B, C, O} \|W - \sum_{k=1}^{K} a_k \odot b_k \odot (c_k + o_k)\|_F^2 + \nu \sum_{n=1}^{N} \|c_n\|_2^2 + \gamma \sum_{n=1}^{N} \|o_n\|_1$$

$$\text{s.t.} \quad A \geq 0, B \geq 0, C \geq 0,$$

$$\sum_{k=1}^{K} a_{nk} = 1, \quad \sum_{k=1}^{K} b_{nk} = 1 \quad \forall n = 1, 2, ..., N$$

where the normalization constraints on the rows of $A$ and $B$ resolve the scalar ambiguity and the $\ell_1$-norm on anomaly factor $o$ promotes sparsity. Efficient solvers for the proposed approach can be developed, and the performance in networks where detection of anomalous nodes such as spammers or intruders is of interest, can be tested. Finally, developing similar approaches for the purpose of anomalous link identification is among our envisioned directions.

5.2.4 Smart egonet-sampling and collapsing over large graphs

One can further capitalize on the robustness provided by the tensor-based toolbox in complex network analysis. Leveraging the sparsity of egonet tensors as well as the parallel implementation of egonet extraction, complexity growth of the proposed DC-EgoTen algorithm remains within affordable limits. In fact, DC-EgoTen has been successfully applied over networks with up to a few million nodes \cite{109}. However, for extremely large networks, where tens to hundreds of millions of nodes interact with each other, the DC-EgoTen algorithm gradually slows down, requiring increased memory and computational capacities. To accommodate analysis of such
humongous networks, one approach is to perform smart-sampling of nodes to construct a reduced-size egonet tensor, namely $\tilde{W} \in \mathbb{R}^{N \times N \times \tilde{N}}$, where $\tilde{N}$ is the number of sampled nodes. The proposed “sketching” will readily result in improved scalability of the algorithm, while the first and second modes of the tensor will enable recovery of ‘community patterns’ via factors $A$ and $B$. Thus, once the PARAFAC decomposition is solved for the subsampled egonet tensor $\tilde{W}$, community association indices of the unsampled nodes can be readily inferred by decomposing their egonet adjacencies on the learned community patterns $a_k \circ b_k^T$ for all $k$.

The performance of the proposed method heavily relies on the node sampling strategy. While the simplest approach is to randomly select $\tilde{N}$ nodes and construct $\tilde{W}$, other ‘smart’ sampling approaches utilizing node centrality or betweenness measures are also viable. In addition, among our proposed approaches is to conduct sampling in a probabilistic way, where the selection probability of a node is proportional to the corresponding absolute value of the (normalized) right-subdominant eigenvector of the adjacency matrix. This metric has documented merits in random-walk literature [75], and its utilization for egonet selection can potentially lead to removing ‘troublesome’ nodes, that is nodes associated with a large number of communities or anomalous nodes, in the construction of a subsampled egonet-tensor.

Furthermore, while sampling egonets provides a natural means of reducing the third dimension of the egonet-tensor, smart summation or ‘collapse’ of different egonet adjacencies is similarly envisioned, and can potentially provide improved performance. In the extreme case, naive summation of the $N$ egonets reduces the three-dimensional egonet tensor to a ‘weighted’ adjacency $\hat{A} := (1/N) \sum_{n=1}^{N} W^{(n)}$, with reinforced structure due to the community structures strengthened via egonet similarities. Subsequently, decomposition of such reinforced adjacency matrix (or the corresponding Laplacian) is proposed to unveil community structure in the network, whose potentially improved performance can be investigated. Similarly, analysis on the effects of such modifications on the spectral properties of the traditional adjacency matrix is among technically challenging yet practically relevant pursuits. Utilization of nodal betweenness and centrality measures as well as the right-subdominant eigenvector of the normalized adjacency matrix are envisioned for this purpose, and their complexity-performance tradeoff is of interest.

### 5.2.5 Ensemble of detection networks against adversarial inputs in CNNs

In Chapter[4] considerable gains in detection of adversarial CNN inputs are effected through careful placement and tuning of the proposed defensive sampling units in Bayesian CNNs. These
results however, also demonstrate that optimal performance against various attack schemes is obtained via different placements of the defensive units. To this end, an interesting direction to extend our current line of work is to consider an ensemble of detection networks, where each network is built by deploying the sampling units at various depths.

Several works have investigated the impact of ensemble learning, and demonstrated remarkable gains in the resultant regression and classification machines [53, p. 605], which motivate well an ensemble of detection networks. However, such advantages require certain assumptions including independence, as well as ‘better-than-random’ performance of each classifier. As such assumptions seem reasonable for the case at hand, extensive empirical tests, as well as theoretical certificates on the achievable improvements in comparison with a single detection network are subject to further investigation, and are of practical importance.
References


Appendix A

Proofs for Chapter 2

A.1 Proof of Proposition 2

The proof of the proposition is inspired by [81] and [78], and is sketched along the following steps.

Step 1. First, we judiciously introduce a surrogate for $F_n(\tilde{L})$ whose minimizer coincides with the SGD updates in (2.13).

To this end, we have that
\[
\min_q f_\nu(x_\nu; \tilde{L}, q) \leq f_\nu(x_\nu; \tilde{L}, q^{[\nu]})
\]

hence,
\[
\hat{F}_n(\tilde{L}) := (1/n) \sum_{\nu=1}^n f_\nu(x_\nu; \tilde{L}, q^{[\nu]})
\]

upper bounds the cost function, namely $F_n(\tilde{L}) \leq \hat{F}_n(\tilde{L})$, $\forall \tilde{L}$. Further approximating $f_n$ through a second-order Taylor’s expansion at the previous subspace update $\tilde{L}[n-1]$, we arrive at

\[
f_n(x_n; \tilde{L}, q[n]) = f_n(x_n; \tilde{L}[n-1], q[n])
\]

\[
+ \text{tr}\{\nabla_{\tilde{L}} f_n(x_n; \tilde{L}[n-1], q[n])(\tilde{L} - \tilde{L}[n-1])^\top\} + \frac{\gamma_n}{2} \|\tilde{L} - \tilde{L}[n-1]\|_H^2.
\]

By choosing $\gamma_n \geq \|\nabla_L^2 f_n(x_n; \tilde{L}[n-1], q[n])\|_H = \|(q[n]q^\top[n]) \otimes I_D + (\lambda/n)I_{rD}\|_H$ and using the norm properties in the Hilbert space, the following can be verified:

(i) $\tilde{f}_n$ is locally tight; i.e., $\tilde{f}_n(x_n; \tilde{L}[n-1], q[n]) = f_n(x_n; \tilde{L}[n-1], q[n]);$

(ii) gradient of $\tilde{f}_n$ is locally tight; i.e., $\nabla_L \tilde{f}_n(x_n; \tilde{L}[n-1], q[n]) = \nabla_L f_n(x_n; \tilde{L}[n-1], q[n]);$

and
(iii) $\tilde{f}_n$ *globally* majorizes the original instantaneous cost $f_n$; that is, $f_n(x_n;\bar{L}, q[n]) \leq \tilde{f}_n(x_n;\bar{L}, q[n]), \forall \bar{L}$.

Selecting now the target surrogate cost as $\tilde{F}_n(\bar{L}) = \frac{1}{n} \sum_{\nu=1}^{n} \tilde{f}_\nu(x_\nu;\bar{L}, q[\nu])$ we have $F_n(\bar{L}) \leq \tilde{F}_n(\bar{L}) \leq \hat{F}_n(\bar{L}), \forall \bar{L}$. Minimizing the cost $\tilde{F}_n(\bar{L})$ amounts to nullifying the gradient, i.e., $\nabla_{\bar{L}} \tilde{F}_n(\bar{L}[n]) = 0$, which yields $\bar{L}[n] = \bar{L}[n-1] - \bar{\gamma}_n^{-1} G_n$, with $\bar{\gamma}_n := \sum_{\nu=1}^{n} \gamma_\nu$. By setting $\mu_n = 1/\bar{\gamma}_n$, the SGD-based update of $\bar{L}[n]$ now coincides with the minimizer of $\tilde{F}_n(\bar{L})$; that is, $\bar{L}[n] = \arg\min_{\bar{L}} \tilde{F}_n(\bar{L})$.

**Step 2.** The second step establishes that the surrogate costs \{\tilde{F}_n(\bar{L})\} form a quasi-martingale sequence [73], and using tightness of the surrogate cost we deduce that $\lim_{n \to \infty} (F_n(\bar{L}[n]) - \tilde{F}_n(\bar{L}[n])) = 0$. Thus, the surrogate cost asymptotically converges to the original cost $F_n(\bar{L})$.

**Step 3.** Leveraging the regularity of $\bar{L}(x_\nu;\bar{L}, q_\nu)$, convergence of the cost sequence implies convergence of $\{\|\nabla_{\bar{L}} F_n(\bar{L}[n]) - \nabla_{\bar{L}} \tilde{F}_n(\bar{L}[n])\|_H\}$ to zero, which along with $\nabla_{\bar{L}} \tilde{F}_n(\bar{L}[n]) = 0$, yields $\{\|\nabla_{\bar{L}} F_n(\bar{L}[n])\|_H\} \to 0$. ■
Appendix B

Proofs for Chapter 4

B.1 Proof of Proposition 1

Define $u_{\nu} := W_2 \sigma(W_1 x_{\nu})$ and approximate it using the first-order Taylor expansion around $\bar{u}_{\nu} := \mathbb{E}_{q_\theta(\omega)}[u_{\nu}]$, to arrive at

$$y_{\nu} \simeq \sigma_{\text{softmax}}(\bar{u}_{\nu}) + \nabla \sigma_{\text{softmax}}(u) \bigg|_{u = \bar{u}_{\nu}} (u_{\nu} - \bar{u}_{\nu}) \quad (B.1)$$

which after taking expectation yields

$$\mathbb{E}_{q_\theta(\omega)}[y_{\nu}] \simeq \sigma_{\text{softmax}}(\bar{u}_{\nu}). \quad (B.2)$$

Upon defining the matrix $H_1 := \nabla \sigma_{\text{softmax}}(u) \bigg|_{u = \bar{u}_{\nu}}$, and using (B.1) and (B.2), we find $y_{\nu} - \mathbb{E}_{q_\theta(\omega)}[y_{\nu}] \simeq H_1 (u_{\nu} - \bar{u}_{\nu})$ that leads to approximating the variance score as

$$\text{Cov}_{q_\theta(\omega)}[y_{\nu}] = \mathbb{E}_{q_\theta(\omega)} \left[ (y_{\nu} - \mathbb{E}_{q_\theta(\omega)}[y_{\nu}]) (y_{\nu} - \mathbb{E}_{q_\theta(\omega)}[y_{\nu}])^\top \right]$$

$$\simeq \mathbb{E}_{q_\theta(\omega)} \left[ H_1 (u_{\nu} - \bar{u}_{\nu}) (u_{\nu} - \bar{u}_{\nu})^\top H_1^\top \right].$$

The trace of the latter can be upper bounded by

$$\text{Tr} \left( \text{Cov}_{q_\theta(\omega)}[y_{\nu}] \right) \leq \lambda_1 \text{Tr} \left( \text{Cov}_{q_\theta(\omega)}[u_{\nu}] \right)$$

108
where $\text{Tr} \left( \text{Cov}_{q_\theta}(\omega)[u_\nu] \right) := \text{Tr} \left( \mathbb{E}_{q_\theta(\omega)}[(u_\nu - \bar{u}_\nu)(u_\nu - \bar{u}_\nu)^\top] \right)$, and $\lambda_1 := \text{Tr}(H_1^\top H_1)$ is deterministic. Thus, the output variance score is upperbounded by that of the previous layer up to a constant $\lambda_1$. Repeating this process of approximating $u_\nu$ as a function of $v_\nu = W_1 x_\nu$ by the first-order Taylor expansion around $v_\nu := \mathbb{E}_{W_1}[v_\nu]$, leads with $H_2 := \nabla \sigma(\bar{v}_\nu)$ to $u_\nu$ and its mean compensated approximation

$$u_\nu \simeq W_2 \sigma(\bar{v}_\nu) + W_2 H_2 (v_\nu - \bar{v}_\nu)$$

(B.3)

The latter yields the covariance approximation

$$\text{Cov}_{q_\theta}(\omega)[u_\nu] \simeq \text{Cov}_{W_2}[W_2 \sigma(\bar{v}_\nu)]$$

$$+ \mathbb{E}_{W_2}[W_2 H_2 \mathbb{E}_{W_1}[(v_\nu - \bar{v}_\nu)(v_\nu - \bar{v}_\nu)^\top] H_2^\top W_2^\top]$$

$$= \mathbb{E}_{W_2}[W_2 \sigma(\bar{v}_\nu)] + \mathbb{E}_{W_2}[W_2 H_2 \mathbb{Cov}_{W_1}[v_\nu] H_2^\top W_2^\top]$$

where we have used the independence of random matrices $W_1$ and $W_2$, and $\mathbb{E}_{W_1}(v_\nu - \bar{v}_\nu) = 0$. Taking the trace and using the inequality $\text{Tr}(AB) \leq \text{Tr}(A)\text{Tr}(B)$ for positive semi-definite matrices $A, B \succeq 0$ twice, we arrive after defining $\lambda_2 := \text{Tr}(H_2^\top H_2)$, at

$$\text{Tr}(\text{Cov}_{q_\theta}(\omega)[u_\nu]) \simeq \text{Cov}_{W_2}[W_2 \sigma(\bar{v}_\nu)] + \mathbb{E}_{W_2}[\text{Tr}(W_2 H_2 \mathbb{Cov}_{W_1}[v_\nu] H_2^\top W_2^\top)]$$

$$\leq \text{Tr}(\text{Cov}_{W_2}[W_2 \sigma(\bar{v}_\nu)]) + \lambda_2 \text{Tr}(\mathbb{E}_{W_2}[W_2 W_2^\top]) \text{Tr}(\text{Cov}_{W_1}[v_\nu]) .$$

Leveraging the last inequality, we can majorize the uncertainty minimization in (4.14) by that in (4.17). This is a coupled minimization of layer-wise variance scores $\text{Tr}(\text{Cov}_{W_1}[v_\nu])$ and $\text{Tr}(\text{Cov}_{W_2}[W_2 \sigma(\bar{v}_\nu)])$, that we solve as follows.

Using $W_1^{\text{TR}}$ along with (4.4) and (4.6), we have $W_1 = W_1^{\text{TR}} S_1 D_1$, where $S_1 := \text{diag}(\{z_{1,1}, z_{1,2}, \cdots, z_{1,h_1}\})$ is the sampling matrix with its pseudo-inverse diagonal mean $D_1 := \text{diag}^\dagger(\mathbb{E}_{q(z,x_1)}[z_{1,1}, z_{1,2}, \cdots, z_{1,h_1}])$. This implies that $v_\nu := \mathbb{E}_{W_1}[v_\nu] = W_1^{\text{TR}} x_\nu$, which does not depend on the sampling vector $p_1$. As a result, the minimization in (4.17) can be readily solved by the proposed subproblems. □
B.2 Proof of Proposition 2

To solve (4.22), consider the Lagrangian

\[ L = \sum_{i=1}^{n} \frac{\alpha_i}{1 - e^{-Cp_i}} + \rho(1^T p - 1) \quad 0 \leq p_i \leq 1 \]

and upon setting its gradient to zero

\[ \frac{\partial L}{\partial p_i} = -C\alpha_i e^{-Cp_i} (1 - e^{-Cp_i})^2 + \rho = 0 \]  \hspace{1cm} (B.4)

and introducing the change of variable

\[ y_i := \exp(-Cp_i) \quad e^{-C} < y_i < 1 \]

we find that (B.4) reduces to

\[ \rho' y_i^2 - (2\rho' + \alpha_i)y_i + \rho' = 0. \]

The feasible root of this quadratic polynomial is

\[ y_i = \frac{2\rho' + \alpha_i - \sqrt{(2\rho' + \alpha_i)^2 - 4\rho'^2}}{2\rho'}. \]

Using the simplex constraint at the optimal point, we find

\[ -\frac{1}{C} \sum_i \ln y_i = 1 \]

which after reverting the change of variable, reduces the optimization in (4.22) to the following root-finding task

\[ \sum_i \ln(2\rho' + \alpha_i - \sqrt{(2\rho' + \alpha_i)^2 - 4\rho'^2}) - n \ln(2\rho') + C = 0. \]

This scalar root-finding problem can be solved using bisection that enjoys super-linear convergence rate. ■
B.3 Selection of defense parameters

In order to further provide insight on the performance against various selection of $B$, $f$, and $\pi$ parameters, Figs. B.1 and B.2 illustrate the AUC-ROC for VM-exact, DVM-log, SAP, and uniform dropout against the combination attack. As the plots suggest, uniform dropout reaches its best performance when placed at the last block, whereas higher performance can be obtained by placing carefully-tuned sampling at units in hidden layers before the last. Furthermore, at a given block $B$, VM-exact demonstrates higher robustness for different values of $f$; that is, smaller fluctuation in AUC is observed, whereas other methods are usually more prone to under-performance given sub-optimal parameters.
Figure B.1: Performance of different sampling mechanisms at various depths and parameters against combination attack on the CIFAR10 dataset
Figure B.2: Performance of different sampling mechanisms at various depths and parameters against combination attack on the cats-and-dogs dataset