My Thesis Title

Diogo Pernes

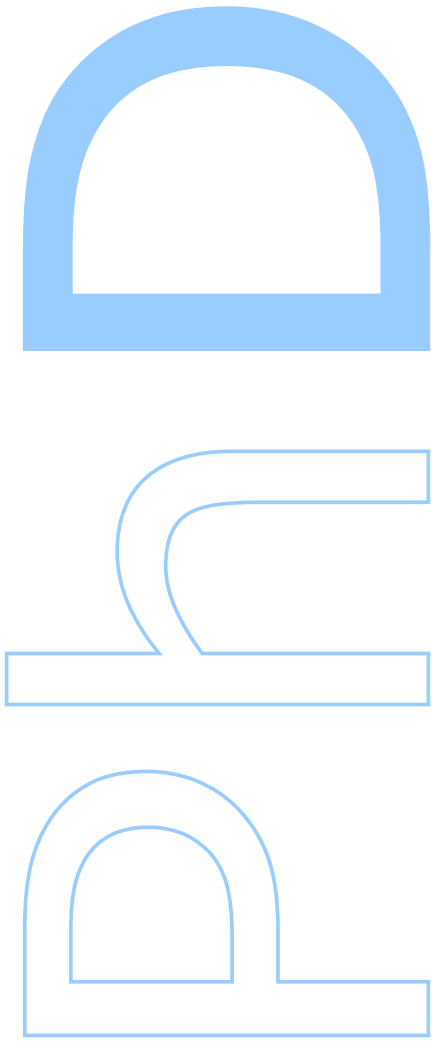
Doutoramento em Ciência de Computadores

Departamento de Ciência de Computadores

2021

Orientador

Jaime S. Cardoso, Professor Catedrático, Faculdade de Engenharia



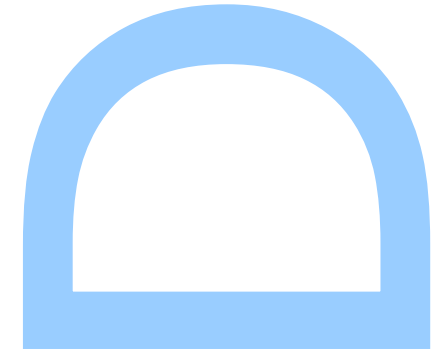
U.PORTO

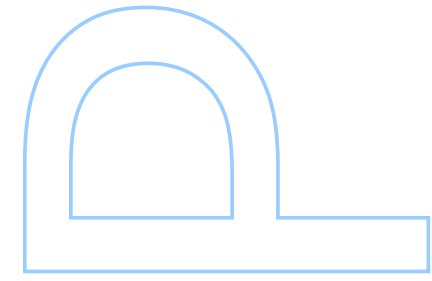
FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Todas as correções determinadas pelo júri, e só essas, foram efetuadas.

O Presidente do Júri,

Porto, _____/_____/_____





UNIVERSIDADE DO PORTO

DOCTORAL THESIS

MyThesis Title

Author:
Diogo PERNES

Supervisor:
Jaime S. CARDOSO

A thesis submitted in fulfilment of the requirements
for the degree of Doctor of Philosophy

at the

Faculdade de Ciências da Universidade do Porto
Departamento de Ciência de Computadores

January 20, 2021

" I am and always will be the optimist, the hoper of far-flung hopes and the dreamer of improbable dreams "

Matt Smith as *The Doctor*, written by Matthew Graham

Acknowledgements

Acknowledge ALL the people!

UNIVERSIDADE DO PORTO

Abstract

Faculdade de Ciências da Universidade do Porto
Departamento de Ciência de Computadores

Doctor of Philosophy

MyThesis Title

by Diogo PERNES

This thesis is about something, I guess.

UNIVERSIDADE DO PORTO

Resumo

Faculdade de Ciências da Universidade do Porto
Departamento de Ciência de Computadores

Doutoramento em Ciência de Computadores

Titulo da Tese em Português

por Diogo PERNES

Este tese é sobre alguma coisa

Contents

List of Figures

Notation and Conventions

In this section, we describe the notation adopted in this thesis. We mostly follow the notation proposed in Goodfellow et al. [1], which is also the recommended one by the International Conference on Learning Representations (ICLR).

Numbers and Arrays

a	A scalar (integer or real)
$\boldsymbol{a}$	A vector
$\boldsymbol{A}$	A matrix
$\mathbf{A}$	A tensor
$\boldsymbol{I}_n$	Identity matrix with n rows and n columns
$\boldsymbol{I}$	Identity matrix with dimensionality implied by context
$\boldsymbol{e}^{(i)}$	Standard basis vector $[0, \dots, 0, 1, 0, \dots, 0]$ with a 1 at position i
$\text{diag}(\boldsymbol{a})$	A square, diagonal matrix with diagonal entries given by $\boldsymbol{a}$
a	A scalar random variable
$\mathbf{a}$	A vector-valued random variable
$\mathbf{A}$	A matrix-valued random variable

Sets and Graphs

$\mathbb{A}$	A set
$\mathbb{R}$	The set of real numbers
$\{0, 1\}$	The set containing 0 and 1
$\{0, 1, \ldots, n\}$	The set of all integers between 0 and n
$[a, b]$	The real interval including a and b
$(a, b]$	The real interval excluding a but including b
$\mathbb{A} \setminus \mathbb{B}$	Set subtraction, i.e., the set containing the elements of $\mathbb{A}$ that are not in $\mathbb{B}$
$\mathcal{G}$	A graph
$Pa_{\mathcal{G}}(\mathrm{x}_i)$	The parents of x_i in $\mathcal{G}$

Indexing

a_i	Element i of vector $\boldsymbol{a}$, with indexing starting at 1
a_{-i}	All elements of vector $\boldsymbol{a}$ except for element i
$A_{i,j}$	Element i, j of matrix $\boldsymbol{A}$
$\boldsymbol{A}_{i,:}$	Row i of matrix $\boldsymbol{A}$
$\boldsymbol{A}_{:,i}$	Column i of matrix $\boldsymbol{A}$
$A_{i,j,k}$	Element (i, j, k) of a 3-D tensor $\mathbf{A}$
$\mathbf{A}_{:,:,i}$	2-D slice of a 3-D tensor
a_i	Element i of the random vector $\mathbf{a}$

Linear Algebra Operations

$\boldsymbol{A}^\top$	Transpose of matrix $\boldsymbol{A}$
$\boldsymbol{A}^+$	Moore-Penrose pseudoinverse of $\boldsymbol{A}$
$\boldsymbol{A} \odot \boldsymbol{B}$	Element-wise (Hadamard) product of $\boldsymbol{A}$ and $\boldsymbol{B}$
$\det(\boldsymbol{A})$	Determinant of $\boldsymbol{A}$

Calculus

$\frac{dy}{dx}$	Derivative of y with respect to x
$\frac{\partial y}{\partial x}$	Partial derivative of y with respect to x
$\nabla_{\boldsymbol{x}} y$	Gradient of y with respect to $\boldsymbol{x}$
$\nabla_{\boldsymbol{X}} y$	Matrix derivatives of y with respect to $\boldsymbol{X}$
$\nabla_{\mathbf{X}} y$	Tensor containing derivatives of y with respect to $\mathbf{X}$
$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}$	Jacobian matrix $\boldsymbol{J} \in \mathbb{R}^{m \times n}$ of $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^m$
$\nabla_{\boldsymbol{x}}^2 f(\boldsymbol{x})$ or $\boldsymbol{H}(f)(\boldsymbol{x})$	The Hessian matrix of f at input point $\boldsymbol{x}$
$\int f(\boldsymbol{x}) d\boldsymbol{x}$	Definite integral over the entire domain of $\boldsymbol{x}$
$\int_{\mathbb{S}} f(\boldsymbol{x}) d\boldsymbol{x}$	Definite integral with respect to $\boldsymbol{x}$ over the set $\mathbb{S}$

Probability and Information Theory

$\mathrm{a} \perp \mathrm{b}$	The random variables a and b are independent
$\mathrm{a} \perp \mathrm{b} \mid \mathrm{c}$	They are conditionally independent given c
$P(\mathrm{a})$	A probability distribution over a discrete variable
$p(\mathrm{a})$	A probability distribution over a continuous variable, or over a variable whose type has not been specified
$\mathrm{a} \sim P$	Random variable a has distribution P
$\mathbb{E}_{\mathrm{x} \sim P}[f(x)]$ or $\mathbb{E}f(x)$	Expectation of $f(x)$ with respect to $P(\mathrm{x})$
$\mathrm{Var}(f(x))$	Variance of $f(x)$ under $P(\mathrm{x})$
$\mathrm{Cov}(f(x), g(x))$	Covariance of $f(x)$ and $g(x)$ under $P(\mathrm{x})$
$H(\mathrm{x})$	Shannon entropy of the random variable x
$D_{\mathrm{KL}}(P \Vert Q)$	Kullback-Leibler divergence of P and Q
$\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$	Gaussian distribution over $\boldsymbol{x}$ with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$

Functions

$f : \mathbb{A} \to \mathbb{B}$	The function f with domain $\mathbb{A}$ and range $\mathbb{B}$
$f \circ g$	Composition of the functions f and g
$f(\boldsymbol{x}; \boldsymbol{\theta})$	A function of $\boldsymbol{x}$ parametrized by $\boldsymbol{\theta}$. (Sometimes we write $f(\boldsymbol{x})$ and omit the argument $\boldsymbol{\theta}$ to lighten notation)
$\log x$	Natural logarithm of x
$\sigma(x)$	Logistic sigmoid, $\dfrac{1}{1+\exp(-x)}$
$\zeta(x)$	Softplus, $\log(1+\exp(x))$
$\|\|\boldsymbol{x}\|\|_p$	L^p norm of $\boldsymbol{x}$
$\|\|\boldsymbol{x}\|\|$	L^2 norm of $\boldsymbol{x}$
x^+	Positive part of x, i.e., $\max(0, x)$
$\mathbf{1}_{\text{condition}}$	is 1 if the condition is true, 0 otherwise

Sometimes we use a function f whose argument is a scalar but apply it to a vector, matrix, or tensor: $f(\boldsymbol{x})$, $f(\boldsymbol{X})$, or $f(\mathbf{X})$. This denotes the application of f to the array element-wise. For example, if $\mathbf{C} = \sigma(\mathbf{X})$, then $C_{i,j,k} = \sigma(X_{i,j,k})$ for all valid values of i, j and k.

Datasets and Distributions

p_{data}	The data generating distribution
$\hat{p}_{\text{data}}$	The empirical distribution defined by the training set
$\mathbb{X}$	A set of training examples
$\boldsymbol{x}^{(i)}$	The i-th example (input) from a dataset
$y^{(i)}$ or $\boldsymbol{y}^{(i)}$	The target associated with $\boldsymbol{x}^{(i)}$ for supervised learning
$\boldsymbol{X}$	The $m \times n$ matrix with input example $\boldsymbol{x}^{(i)}$ in row $\boldsymbol{X}_{i,:}$

Chapter 1

Background

In this chapter, we provide a brief overview of the main methods we are going to use.

Chapter 2

Networked data streams

In this chapter, we present our results in networked data streams.

2.1 SpaMHMM: Sparse Mixture of Hidden Markov Models

The proposed model finds intersections with distributed sparse representations.

2.1.1 Overview

Sparse representation/coding expresses a signal/model f, defined over some independent variable x, as a linear combination of a few atoms from a prespecified and overcomplete dictionary of size M:

$$f(x) = \sum_{m=1}^{M} s_m \phi_m(x), \tag{2.1}$$

where $\phi_m(x)$ are the atoms and only a few of the scalars s_m are non-zero, providing a sparse representation of $f(x)$.

Distributed sparse representation [2] is an extension of the standard version that considers networks with K nodes. At each node, the signal sensed at the same node has its sparsity property because of its intracorrelation, while, for networks with multiple nodes, signals received at different nodes also exhibit strong intercorrelation. The intra- and inter-correlations lead to a joint sparse model. An interesting scenario in distributed sparse representation is when all signals/models share the common support but with different non-zero coefficients.

Inspired by the formulation of equation (2.1), we propose to model the generative distribution of the data coming from each of the K nodes of a network as a sparse mixture

obtained from a dictionary of generative distributions. Specifically, we shall model the distribution for each node as a sparse mixture over a 'large' shared dictionary of HMMs, where each HMM corresponds to an individual atom from the dictionary. The field knowledge about the similarities between nodes is summarized in an affinity matrix. The objective function of the learning process promotes reusing HMM atoms between similar nodes. We now formalize these ideas.

2.1.2 Model formulation

2.1.2.1 Definition

Assume we have a set of nodes $\mathbb{Y} = \{1, ..., K\}$ connected by an undirected weighted graph $\mathcal{G}$, expressed by a symmetric matrix $\boldsymbol{G} \in \mathbb{R}^{K \times K}$. These nodes thus form a network, in which the weights are assumed to represent degrees of affinity between each pair of nodes (i.e. the greater the edge weight, the more the respective nodes *like* to agree). The nodes y in the graph produce D-dimensional sequences $\mathbf{X} = \left(\mathbf{x}^{(1)}, ..., \mathbf{x}^{(T)}\right)$, $\mathbf{x}^{(t)} \in \mathbb{R}^D$, whose conditional distribution we shall model using a mixture of HMMs:

$$p(\mathbf{X} \mid \mathrm{y}) = \sum_{\mathrm{z}} p(\mathrm{z} \mid \mathrm{y}) p(\mathbf{X} \mid \mathrm{z}), \tag{2.2}$$

where $\mathrm{z} \in \{1, ..., M\}$ is a latent random variable, being M the size of the mixture. This is a particular realization of equation (2.1) where f is the probability density function $p(\mathbf{X} \mid \mathrm{y})$ and the coefficients s_m correspond to the probabilities $p(\mathrm{z} = m \mid \mathrm{y})$. Here, $p(\mathbf{X} \mid \mathrm{z})$ is the marginal distribution of observations of a standard first-order homogeneous HMM:

$$p(\mathbf{X} \mid \mathrm{z}) = \sum_{\mathbf{h}} p(\mathrm{h}^{(0)} \mid \mathrm{z}) \prod_t p(\mathrm{h}^{(t)} \mid \mathrm{h}^{(t-1)}, \mathrm{z}) p(\mathbf{x}^{(t)} \mid \mathrm{h}^{(t)}, \mathrm{z}), \tag{2.3}$$

where $\mathbf{h} = \left(\mathrm{h}^{(0)}, ..., \mathrm{h}^{(T)}\right)$, $\mathrm{h}^{(t)} \in \{1, ..., S\}$, is the sequence of hidden states of the HMM, being S the number of hidden states. Note that the factorization in equation (2.2) imposes conditional independence between the sequence $\mathbf{X}$ and the node y, given the latent variable z. This is a key assumption of this model, since this way the distributions for the observations in the nodes in $\mathbb{Y}$ share the same dictionary of HMMs, promoting parameter sharing among the K mixtures.

2.1.2.2 Inference

Given an observed sequence $\boldsymbol{X}$ and its corresponding node $y \in \mathbb{Y}$, the inference problem here consists in finding the likelihood $p(\mathbf{X} = \boldsymbol{X} \mid \mathrm{y} = y)$ (from now on, abbreviated as $p(\boldsymbol{X} \mid y)$) as defined by equations (2.2) and (2.3). The marginals $p(\boldsymbol{X} \mid \mathrm{z})$ of each HMM in the mixture may be computed efficiently, in $O(S^2T)$ time, using the Forward algorithm [3]. Then, $p(\boldsymbol{X} \mid y)$ is obtained by applying equation (2.2), so inference in the overall model is done in at most $O(MS^2T)$ time. As we shall see, however, the mixtures we get after learning will often be sparse (see Section 2.1.2.3), leading to an even smaller time complexity.

2.1.2.3 Learning

Given an i.i.d. dataset consisting of N tuples $(\boldsymbol{X}_i, y_i)$ of sequences of observations $\boldsymbol{X}_i = \left(\boldsymbol{x}_i^{(1)}, ..., \boldsymbol{x}_i^{(T_i)}\right)$ and their respective nodes $y_i \in \mathbb{Y}$, the model defined by equations (2.2) and (2.3) may be easily trained using the Expectation-Maximization (EM) algorithm [4], (locally) maximizing the usual log-likelihood objective:

$$J(\theta) = \sum_{i=1}^{N} \log p(\boldsymbol{X}_i \mid y_i, \theta), \tag{2.4}$$

where θ represents all model parameters, namely:

1. the M-dimensional mixture coefficients, $\boldsymbol{\alpha}_k \coloneqq \big(p(\mathrm{z} = 1 \mid \mathrm{y} = k), ..., p(\mathrm{z} = M \mid \mathrm{y} = k)\big)$, for $k = 1, ..., K$;
2. the S-dimensional initial state probabilities, $\boldsymbol{\pi}_m \coloneqq \Big(p(\mathrm{h}^{(0)} = 1 \mid \mathrm{z} = m), ..., p(\mathrm{h}^{(0)} = S \mid \mathrm{z} = m)\Big)$, for $m = 1, ..., M$;
3. the $S \times S$ state transition matrices, $\boldsymbol{A}^m$, where $A^m_{s,u} \coloneqq p(\mathrm{h}^{(t)} = u \mid \mathrm{h}^{(t-1)} = s, \mathrm{z} = m)$, for $s, u = 1, ..., S$ and $m = 1, ..., M$;
4. the emission probability means, $\boldsymbol{\mu}_{m,s} \in \mathbb{R}^D$, for $m = 1, ..., M$ and $s = 1, ..., S$;
5. the emission probability diagonal covariance matrices, $\boldsymbol{I}\boldsymbol{\sigma}^2_{m,s}$, where $\boldsymbol{\sigma}^2_{m,s} \in \mathbb{R}^D_+$, for $m = 1, ..., M$ and $s = 1, ..., S$.

Here, we are assuming that the emission probabilities $p(\mathrm{x}^{(t)} \mid \mathrm{h}^{(t)}, \mathrm{z})$ are Gaussian with diagonal covariances. This introduces almost no loss of generality, since the extension of

this work to discrete observations or other types of continuous emission distributions is straightforward.

The procedure to maximize objective (2.4) using EM is described in Algorithm 1. The update formulas follow from the standard EM procedure and can be obtained by viewing this model as a Bayesian network or by following the derivation detailed in Section **??**. However, the objective (2.4) does not take advantage of the known structure of $\mathcal{G}$. In order to exploit this information, we introduce a regularization term, maximizing the following objective instead:

$$\begin{aligned} J_r(\theta) &= \frac{1}{N}\sum_{i=1}^{N}\log p(\boldsymbol{X}_i \mid y_i, \theta) \\ &\quad + \frac{\lambda}{2}\sum_{\substack{j,k=1,\\ k\neq j}}^{K} G_{j,k}\mathbb{E}_{\mathrm{z}\sim p(\mathrm{z}\mid \mathrm{y}=j,\theta)}[p(\mathrm{z}\mid \mathrm{y}=k,\theta)] \\ &= \frac{1}{N}\sum_{i=1}^{N}\log p(\boldsymbol{X}_i \mid y_i, \theta) + \frac{\lambda}{2}\sum_{\substack{j,k=1,\\ k\neq j}}^{K} G_{j,k}\boldsymbol{\alpha}_j^{\top}\boldsymbol{\alpha}_k, \end{aligned} \tag{2.5}$$

where $\lambda \geq 0$ controls the relative weight of the two terms in the objective. Note that this regularization term favors nodes connected by edges with large positive weights to have similar mixture coefficients and thus share mixture components. On the other hand, nodes connected by edges with large negative weights will tend to have orthogonal mixture coefficients, being described by disjoint sets of components. These observations agree with our prior assumption that the edge weights express degrees of similarity between each pair of nodes. Proposition 2.1 formalizes these statements and enlightens interesting properties about the expectations $\mathbb{E}_{\mathrm{z}\sim p(\mathrm{z}\mid \mathrm{y}=j,\theta)}[p(\mathrm{z}\mid \mathrm{y}=k,\theta)]$.

Proposition 2.1. *Let $\mathbb{P}_M$ be the set of all M-nomial probability distributions and $M > 1$. We have:*

1. $\min_{p,q\in\mathbb{P}_M}\mathbb{E}_{\mathrm{z}\sim p}[q(\mathrm{z})] = 0$;
2. $\arg\min_{p,q\in\mathbb{P}_M}\mathbb{E}_{\mathrm{z}\sim p}[q(\mathrm{z})] = \{p, q \in \mathbb{P}_M \mid \forall\, m \in \{1,...,M\} : p(\mathrm{z}=m)q(\mathrm{z}=m) = 0\}$;
3. $\max_{p,q\in\mathbb{P}_M}\mathbb{E}_{\mathrm{z}\sim p}[q(\mathrm{z})] = 1$;
4. $\arg\max_{p,q\in\mathbb{P}_M}\mathbb{E}_{\mathrm{z}\sim p}[q(\mathrm{z})] = \{p, q \in \mathbb{P}_M \mid \exists\, m \in \{1,...,M\} : p(\mathrm{z}=m) = q(\mathrm{z}=m) = 1\}$.

Proof. By the definition of expectation,

$$\mathbb{E}_{z\sim p}[q(z)] = \sum_{m=1}^{M} p(z=m)q(z=m). \tag{2.6}$$

Statements 1 and 2 follow immediately from the fact that every term in the right-hand side of (2.6) is non-negative and $M > 1$. For the remaining, we rewrite (2.6) as the dot product of two M-dimensional vectors $\boldsymbol{\alpha}_p$ and $\boldsymbol{\alpha}_q$, representing the two distributions p and q, respectively, and we use the following linear algebra inequalities to build an upper bound for this expectation:

$$\mathbb{E}_{z\sim p}[q(z)] = \boldsymbol{\alpha}_p^\top \boldsymbol{\alpha}_q \leq ||\boldsymbol{\alpha}_p||_2||\boldsymbol{\alpha}_q||_2 \leq ||\boldsymbol{\alpha}_p||_1||\boldsymbol{\alpha}_q||_1 = 1, \tag{2.7}$$

where $||\cdot||_1$ and $||\cdot||_2$ are the L^1 and L^2 norms, respectively. Clearly, the equality $\mathbb{E}_{z\sim p}[q(z)] = 1$ holds if p and q are chosen from the set defined in statement 4, where the distributions p and q are the same and they are non-zero for a single assignment of z. This proves statement 3. Now, to prove statement 4, it suffices to show that there are no other maximizers. The first inequality in (2.7) is transformed into an equality if and only if $\boldsymbol{\alpha}_p = \boldsymbol{\alpha}_q$, which means $p \equiv q$. The second inequality becomes an equality when the L^1 and L^2 norms of the vectors coincide, which happens if and only if the vectors have only one non-zero component, concluding the proof. □

Specifically, given two distinct nodes $j, k \in \mathbb{Y}$, if $G_{j,k} > 0$, the regularization term for these nodes is maximum (and equal to $G_{j,k}$) when the mixtures for these two nodes are the same and have one single active component (i.e. one mixture component whose coefficient is non-zero). On the contrary, if $G_{j,k} < 0$, the term is maximized (and equal to zero) when the mixtures for the two nodes do not share any active components. In both cases, though, we conclude from Proposition 2.1 that we are favoring sparse mixtures. We see sparsity as an important feature since it allows the size M of the dictionary of models to be large and therefore expressive without compromising our rational that the observations in a given node are well modeled by a mixture of only a few HMMs. This way, some components will specialize on describing the behavior of some nodes, while others will specialize on different nodes. Moreover, sparse mixtures yield faster inference, more interpretable models and (possibly) less overfitting. By setting $\lambda = 0$, we clearly get the initial objective (2.4), where inter-node correlations are modeled only via parameter sharing. As $\lambda \to \infty$, two interesting scenarios may be anticipated. If $G_{j,k} > 0$, $\forall j, k \in \mathbb{Y}$, all nodes will tend do share the same single mixture component, i.e. we would be learning

one single HMM to describe the whole network. If $G_{j,k} < 0$, $\forall j,k \in \mathbb{Y}$, and $M \geq K$, each node would tend to learn its own HMM model independently from all the others. Again, in both scenarios, the obtained mixtures are sparse.

The objective function (2.5) can still be maximized via EM (see details in Section **??**). However, the introduction of the regularization term in the objective makes it impossible to find a closed form solution for the update formula of the mixture coefficients. Thus, in the M-step, we need to resort to gradient ascent to update these parameters. In order to ensure that the gradient ascent iterative steps lead to admissible solutions, we adopt the following reparameterization from [5]:

$$\alpha_{k,m} = \frac{\sigma\left(\beta_{k,m}\right)^2}{\sum_{l=1}^{M} \sigma\left(\beta_{k,l}\right)^2}, \tag{2.8}$$

for $k = 1, ..., K$ and $m = 1, ..., M$, and where $\sigma(\cdot)$ is the rectifier linear (ReLU) function. This reparameterization clearly resembles the softmax function, but, contrarily to that one, admits sparse outputs. The squared terms in equation (2.8) aim only to make the optimization more stable. The optimization steps for the objective (2.5) using this reparameterization are described in Algorithm 2.

Algorithm 1 EM algorithm for the mixture without regularization (MHMM).

Inputs: The training set, consisting of N tuples $(\boldsymbol{X}_i, y_i)$, a set of initial parameters $\theta^{(0)}$ and the number of training iterations $\mathcal{I}$.
for $j = 1, ..., \mathcal{I}$ **do**
Sufficient statistics:

1. $n_k \coloneqq \sum_i \mathbf{1}_{y_i=k}$, where $\mathbf{1}_{(\cdot)}$ is the indicator function, for $k = 1, ..., K$.
2. Obtain the mixture posteriors $\eta_{i,m} \coloneqq p(\mathrm{z} = m|\boldsymbol{X}_i, y_i, \theta^{(j-1)})$, for $i = 1, ..., N$ and $m = 1, ..., M$, by computing $\tilde{\eta}_{i,m} \coloneqq p(\boldsymbol{X}_i|\mathrm{z} = m, \theta^{(j-1)})p(\mathrm{z} = m|y_i, \theta^{(j-1)})$ and normalizing it.
3. Obtain the state posteriors $\gamma_{i,m,s}(t) \coloneqq p(\mathrm{h}^{(t)} = s|\mathrm{z} = m, \boldsymbol{X}_i, \theta^{(j-1)})$ and $\xi_{i,m,s,u}(t) \coloneqq p(\mathrm{h}^{(t-1)} = s, \mathrm{h}^{(t)} = u|\mathrm{z} = m, \boldsymbol{X}_i, \theta^{(j-1)})$, for $i = 1, ..., N$, $m = 1, ..., M$ and $s, u = 1, ..., S$, as done in the Baum-Welch algorithm [6].

M-step:

1. $\alpha_{k,m} = \frac{\sum_i \eta_{i,m}\mathbf{1}_{y_i=k}}{n_k}$, for $k = 1, ..., K$ and $m = 1, ..., M$, obtaining $\boldsymbol{\alpha}_k$.
2. $\pi_{m,s} = \frac{\sum_i \eta_{i,m}\gamma_{i,m,s}(0)}{\sum_i \eta_{i,m}}$, for $m = 1, ..., M$ and $s = 1, ..., S$, obtaining $\boldsymbol{\pi}_m$.
3. $A^m_{s,u} = \frac{\sum_i \eta_{i,m} \sum_{t=1}^{T_i} \xi_{i,m,s,u}(t)}{\sum_i \eta_{i,m} \sum_{t=0}^{T_i-1} \gamma_{i,m,s}(t)}$, for $m = 1, ..., M$ and $s, u = 1, ..., S$, obtaining $\boldsymbol{A}^m$.
4. $\boldsymbol{\mu}_{m,s} = \frac{\sum_i \eta_{i,m} \sum_{t=1}^{T_i} \gamma_{i,m,s}(t)\boldsymbol{x}_i^{(t)}}{\sum_i \eta_{i,m} \sum_{t=1}^{T_i} \gamma_{i,m,s}(t)}$, for $m = 1, ..., M$ and $s = 1, ..., S$.
5. $\boldsymbol{\sigma}^2_{m,s} = \frac{\sum_i \eta_{i,m} \sum_{t=1}^{T_i} \gamma_{i,m,s}(t)\left(\boldsymbol{x}_i^{(t)} - \boldsymbol{\mu}_s^m\right)^2}{\sum_i \eta_{i,m} \sum_{t=1}^{T_i} \gamma_{i,m,s}(t)}$, for $m = 1, ..., M$ and $s = 1, ..., S$.
6. $\theta^{(j)} = \bigcup_{k,m,s} \left\{\boldsymbol{\alpha}_k, \boldsymbol{\pi}_m, \boldsymbol{A}^m, \boldsymbol{\mu}_{m,s}, \boldsymbol{\sigma}^2_{m,s}\right\}$.

end for

Algorithm 2 EM algorithm for the mixture with regularization (SpaMHMM).

Inputs: The training set, consisting of N tuples $(\boldsymbol{X}_i, y_i)$, the matrix $\boldsymbol{G}$ describing the graph $\mathcal{G}$, the regularization hyperparameter λ, a set of initial parameters $\theta^{(0)}$, the number of training iterations $\mathcal{I}$, the number of gradient ascent iterations $\mathcal{J}$ to perform on each M-step, the learning rate ρ for the gradient ascent.

for $j = 1, ..., \mathcal{I}$ **do**

Sufficient statistics: same as in Algorithm 1.

M-step:

for $l = 1, ..., \mathcal{J}$ **do**

1. $\psi_{k,m} := \frac{1}{N} \sum_i (\eta_{i,m} - \alpha_{k,m}) \mathbf{1}_{y_i = k}$, for $k = 1, ..., K$ and $m = 1, ..., M$.
2. $\omega_{k,m} := \alpha_{k,m} \sum_{j \neq k} G_{j,k} \left(\alpha_{j,m} - \boldsymbol{\alpha}_j^\top \boldsymbol{\alpha}_k \right)$, for $k = 1, ..., K$ and $m = 1, ..., M$.
3. $\delta_{k,m} := \mathbf{1}_{\beta_{k,m} > 0} \frac{2\sigma'(\beta_{k,m})}{\sigma(\beta_{k,m})} \left(\psi_{k,m} + \lambda \omega_{k,m} \right)$, where $\sigma'(\cdot)$ is the derivative of $\sigma(\cdot)$, for $k = 1, ..., K$ and $m = 1, ..., M$.
4. $\beta_{k,m} \leftarrow \beta_{k,m} + \rho \delta_{k,m}$, for $k = 1, ..., K$ and $m = 1, ..., M$.
5. Use equation (2.8) to obtain $\alpha_{k,m}$, for $k = 1, ..., K$ and $m = 1, ..., M$.

end for

Do steps 2) – 6) in the M-step of Algorithm 1.

end for

2.2 Experimental Evaluation

The model was developed on top of the library hmmlearn [7] for Python, which implements inference and unsupervised learning for the standard HMM using a wide variety of emission distributions. Both learning and inference use the hmmlearn API, with the appropriate adjustments for our models. For reproducibility purposes, we make our source code, pre-trained models and the datasets publicly available [*].

We evaluate four different models in our experiments: a model consisting of a single HMM (denoted as 1-HMM) trained on sequences from all graph nodes; a model consisting of K HMMs trained independently (denoted as K-HMM), one for each graph node; a mixture of HMMs (denoted as MHMM) as defined in this work (equations (2.2) and (2.3)), trained to maximize the usual log-likelihood objective (2.4); a mixture of HMMs (denoted as SpaMHMM) as the previous one, trained to maximize our regularized objective (2.5).

Models 1-HMM, K-HMM and MHMM will be our baselines. We shall compare the performance of these models with that of SpaMHMM and, for the case of MHMM, we shall also verify if SpaMHMM actually produces sparser mixtures in general, as argued in Section 2.1.2.3. In order to ensure a fair comparison, we train models with approximately the same number of possible state transitions. Hence, given an MHMM or SpaMHMM with M mixture components and S states per component, we train a 1-HMM with $\approx S\sqrt{M}$ states and a K-HMM with $\approx S\sqrt{M/K}$ states per HMM. We initialize the mixture coefficients in MHMM and SpaMHMM randomly, while the state transition matrices and the initial state probabilities are initialized uniformly. Means are initialized using k-means, with k equal to the number of hidden states in the HMM, and covariances are initialized with the diagonal of the training data covariance. Models 1-HMM and K-HMM are trained using the Baum-Welch algorithm, MHMM is trained using Algorithm 1 and SpaMHMM is trained using Algorithm 2. However, we opted to use Adam [8] instead of *vanilla* gradient ascent in the inner loop of Algorithm 2, since its per-parameter learning rate proved to be beneficial for faster convergence.

2.2.1 Anomaly detection in Wi-Fi networks

A typical Wi-Fi network infrastructure is constituted by K access points (APs) distributed in a given space. The network users may alternate between these APs seamlessly, usually connecting to the closest one. There is a wide variety of anomalies that may happen

[*] https://github.com/dpernes/spamhmm

during the operation of such network and their automatic detection is, therefore, of great importance for future mitigation plans. Some anomalous behaviors are: overloaded APs, failed or crashed APs, persistent radio frequency interference between adjacent APs, authentication failures, etc. However, obtaining reliable ground truth annotation of these anomalies in entire wireless networks is costly and time consuming. Under these circumstances, using data obtained through realistic network simulations is a common practice.

In order to evaluate our model in the aforementioned scenario, we have followed the procedure of [9], performing extensive network simulations in a typical Wi-Fi network setup (IEEE 802.11 WLANg 2.4 GHz in infrastructure mode) using OMNeT++ [10] and INET [11] simulators. Our network consists of 10 APs and 100 users accessing it. The pairwise distances between APs are known and fixed. Each sequence contains information about the traffic in a given AP during 10 consecutive hours and is divided in time slots of 15 minutes without overlap. Thus, every sequence has the same length, which is equal to 40 samples (time slots). Each sample contains the following 7 features: the number of unique users connected to the AP, the number of sessions within the AP, the total duration (in seconds) of association time of all current users, the number of octets transmitted and received in the AP and the number of packets transmitted and received in the AP. Anomalies typically occur for a limited amount of time within the whole sequence. However, in this experiment, we label a sequence as "anomalous" if there is at least one anomaly period in the sequence and we label it as "normal" otherwise. One of the simulations includes normal data only, while the remaining include both normal and anomalous sequences. In order to avoid contamination of normal data with anomalies that may occur simultaneously in other APs, we used the data of the normal simulation for training (150 sequences) and the remaining data for testing (378 normal and 42 anomalous sequences).

In a Wi-Fi network, as users move in the covered area, they disconnect from one AP and they immediately connect to another in the vicinity. As such, the traffic in adjacent APs may be expected to be similar. Following this idea, the weight $G_{j,k}$, associated with the edge connecting nodes j and k in graph $\mathcal{G}$, was set to the inverse distance between APs j and k and normalized so that $\max_{j,k} G_{j,k} = 1$. As in [9], sequences were preprocessed by subtracting the mean and dividing by the standard deviation and applying PCA, reducing the number of features to 3. For MHMM, we did 3-fold cross validation of the number of mixture components M and hidden states per component S. We ended up using $M = 15$ and $S = 10$. We then used the same values of M and S for SpaMHMM and we did 3-fold

cross validation for the regularization hyperparameter λ in the range $[10^{-4}, 1]$. The value $\lambda = 10^{-1}$ was chosen. We also cross-validated the number of hidden states in 1-HMM and K-HMM around the values indicated in Section 2.2. Every model was trained for 100 EM iterations or until the loss plateaus. For SpaMHMM, we did 100 iterations of the inner loop on each M-step, using a learning rate $\rho = 10^{-3}$. We repeat training 10 times for each model, starting from different random initializations, in order to reduce the likelihood of erroneous results due to local minima trapping.

Models were evaluated by computing the average log-likelihood per sample on normal and anomalous test data, plotting the receiver operating characteristic (ROC) curves and computing the respective areas under the curves (AUCs). The small standard deviations in Table **??** attest the robustness of the adopted initialization scheme and learning algorithms. Figure **??** shows that the ROC curves for MHMM and SpaMHMM are very similar and that these models clearly outperform 1-HMM and K-HMM. This is confirmed by the AUC and log-likelihood results in Table **??**. Although K-HMM achieved the best (lowest) average log-likelihood on anomalous data, this result is not relevant, since it also achieved the worst (lowest) average log-likelihood on normal data. This is in fact the model with the worst performance, as shown by its ROC and respective AUC.

Appendix A

Appendix Title Here

Write your Appendix content here.

Bibliography

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org. [Cited on page xv.]

[2] D. Baron, M. F. Duarte, M. B. Wakin, S. Sarvotham, and R. G. Baraniuk, "Distributed compressive sensing," *CoRR*, vol. abs/0901.3403, 2009. [Online]. Available: http://arxiv.org/abs/0901.3403 [Cited on page 3.]

[3] L. R. Rabiner and B.-H. Juang, "An introduction to hidden markov models," *ieee assp magazine*, vol. 3, no. 1, pp. 4–16, 1986. [Cited on page 5.]

[4] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the royal statistical society. Series B (methodological)*, pp. 1–38, 1977. [Cited on page 5.]

[5] Z. Yang, J. Zhao, B. Dhingra, K. He, W. W. Cohen, R. Salakhutdinov, and Y. LeCun, "Glomo: Unsupervisedly learned relational graphs as transferable representations," 2018. [Online]. Available: https://arxiv.org/abs/1806.05662 [Cited on page 8.]

[6] L. Baum, "An inequality and associated maximization technique in statistical estimation of probabilistic functions of a markov process," *Inequalities*, vol. 3, pp. 1–8, 1972. [Cited on page 9.]

[7] S. Lebedev. hmmlearn, hidden markov models in python, with scikit-learn like api. [Online]. Available: https://github.com/hmmlearn/hmmlearn [Cited on page 11.]

[8] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014. [Cited on page 11.]

[9] A. Allahdadi, R. Morla, and J. S. Cardoso, "802.11 wireless simulation and anomaly detection using HMM and UBM," *CoRR*, vol. abs/1707.02933, 2017. [Online]. Available: http://arxiv.org/abs/1707.02933 [Cited on page 12.]

[10] Omnet++, discrete event simulator. [Online]. Available: https://www.omnetpp.org/ [Cited on page 12.]

[11] Inet framework. [Online]. Available: https://inet.omnetpp.org/ [Cited on page 12.]