Thèse de Matteo Causo, Lille 1, 2017

Neuro-Inspired Energy-Efficient Computing Platforms

THÈSE No 1234 (2016)

PRÉSENTÉE LE 4 JANVIER 2017 À

L'Université de Lille 1
Ecole Doctorale Sciences pour L'Ingénieur

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Matteo CAUSO

acceptée sur proposition du jury :

Edith Beigné, CEA-LETI	rapporteur
Marian Verhelst, KU Leuven	rapporteur
Lirida Alves de Barros Naviner, Telecom ParisTech	examinateur
Laurent Clavier, Université Lille 1	président
Antoine Frappé, ISEN-IEMN Lille	examinateur
Andreia Cathelin, STMicroelectronics	co-encadrant et tuteur industriel
Jan M. Rabaey, UC Berkeley	co-encadrant scientifique
Andreas Kaiser, ISEN-IEMN Lille	directeur de thèse
Pascal Yim, EC Lille	invité

Lille, France
2016

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

To Nonna Tetta

A: "Nonna Anna, chi è quella vecchietta nella foto con papà?"

B: "Quella è la mamma mia, la nonna Tetta, la nonna di papà! Era vecchietta, ma ne faceva di cose! La nonna Tetta andava a cavallo, e ha fatto pure tante gare! Gestiva il calcio a Casarano e la sera andava in discoteca, ma alla fine si è sposata! Ha lavorato a lungo in ferrovia, poi ha aperto un'azienda. Ma non una sola, almeno due. Pensa te che tutta stanca, si svegliava alle 5 per fare pasticciotti. O almeno, lei diceva così... I pasticciotti, sì! Ma faceva tante altre cose. È che fatica impastare e guidare su e giù per tutta la Puglia. É andata a Milano, la nonna Tetta, e per due volte! Prima ha fatto un corso di chimica, poi ha parlato l'inglese a l'Expo. Sì, la nonna Tetta conosceva bene l'inglese! Lo ha imparato quando è stata in Irlanda, Inghilterra, America e in Australia. Là si è divertita tanto, dovevi vedere che sorriso che aveva! Così contenta l'ho vista tante altre volte, per la verità! Per esempio quando è diventata ragioniera, ingegnera o economista. Voleva pure fare l'avvocato! Per un bel po' ha vissuto a Torino. Là la conoscevano tutti, e pure là le volevano tanto bene. Poi è partita per la Spagna ed ora parla lo spagnolo. Sì, ma non solo! É emigrata in Francia per fare cose strane, e ha dovuto imparare anche il francese, altrimenti nessuno la capiva! Quando la domenica usciva da Notre Dame, dove si è cresimata, dopo la messa andava a comprare sempre il suo dolce preferito e un sablé semplice semplice. Pensa te quanto era particolare la nonna Tetta: ha chiamato la sua pasticceria proprio Sablé un nome semplice semplice. Là dentro faceva magie ad opera d'arte: panettoni, uova di Pasqua giganti, paste di mandorla e persino torte straniere, Cubane! Ma il giovedì non la trovi... Lei doveva andare a pesca! Sì, perché lei si divertiva a pescare. Poi il pesce lo vendeva congelato. Dovevi vederla con l'uniforme, tutta bella e organizzata nel negozio tutto nuovo, con tante idee e tanti colori studiati. Ogni volta che andavamo ci offriva caffè e dolcetti da provare. Come cucinava bene, la nonna Tetta! Faceva la salsa e il pane l'estate in campagna e per ferragosto organizzava una grande festa. C'era sempre tanto da festeggiare! Aveva appena finito casa e si poteva sposare. Sì, la nonna Tetta costruiva le case. Dovevi vederla col camioncino blu o con la Passat verde quante feste ti faceva quando la incrociavi per strada. Si commoveva facilmente la nonna Tetta. Ma che vuoi, era appena diventata mamma tante volte. Pensa te, una volta ha fatto tre figli in una volta! Quanti sacrifici che ha dovuto fare, quanto è stato difficile, ma alla fine ha riempito di gioia tutta la casa e che da farsi! Aveva appena fatto gli esami di maturità, quelli finali delle medie e stava studiando il latino. Aveva lavorato per tanti anni per le assicurazioni, era estetista ma le piaceva tanto danzare a tutti i livelli e tanti tipi di ballo. Alle feste non la fermavi, balli di gruppo e valzer erano la sua specialità. Però la sera si addormentava dappertutto, era stanca la nonna Tetta! La mattina in ufficio, toccava a lei! Tutti i giorni andava a Maglie e al Comune. Piaceva tanto il caffè alla nonna Tetta e fumava, pure troppo! La nonna Tetta era il Natale da lei, Padre Pio alla TV e il suo compleanno festeggiato il 12 giugno."

A: "Nonna Anna, sarò pure piccolo ma non mi prendi in giro: non ha senso quello che hai detto! Ha detto la mamma che la nonna Tetta era Amore, è questo che vuoi dire?! Ho capito... È l'Amore di quando papà vi abbraccia tutti dopo tanto tempo che non vi vede. Questo è la nonna Tetta, grazie nonna Anna!"

© 2017 Tous droits réservés.

Acknowledgements

Firstly, I want to thank my academic advisors Prof. Jan Rabaey, UC Berkeley, Prof. Andreas Kaiser and Prof. Antoine Frappé, ISEN-Lille and my industrial supervisor HDR Andreia Cathelin, STMicroelectronics Crolles. It has been an honor to be part of their team and to actively contribute to the success of this international collaboration.

In fact, every productive and stimulating experience and even the difficulty, such a big project involved, finally built an objective-oriented, fast-failure, and extremely open-minded and dynamic research culture of mine. Through the challenges they provided me with, I matured both personally and professionally. Furthermore, the necessity to be integrated in different teams in Berkeley, California, in Lille and in Crolles, France contributed in reinforcing my educational path already oriented to the internationalization and enriched by diverse cultural backgrounds.

My time at Berkeley, Lille and Grenoble/Crolles was made enjoyable in large part due to the many friends that became a part of my life. I am grateful to Antonio Puglielli, Simone Benatti, Abbas Rahimi, Lorenzo Iotti, Ali Moin, Katerina Papadopoulou, Olivia Nolan, Candy Corpus, Brian Zimmer, James Dunn, Brian Richards, Sameet Ramakrishnan, Krishna Settaluri, Bo Zhao, Stevo Bailey, Leslie Nishiyama; Cristian Marin, Ilias Sourikopoulos, Florence Alberti; Martin Cochet, Artiom Tchouprina, Etienne Marecal, Nicolas Le Dortz, Roberto Guizzetti, Dimitri Gaughet, Abhirup Lahiri, Carlo Tinella, Emmanuel Rouat, Pascal Urard, Philip Cathelin, Gabriel Mugny, Reda Kasri, Hanae Zegmout, Hani Malloug, Hani Sherry, Julien Kieffer, Florian Voineau, Pascale Maillet-Contoz.

Lastly, I would like to thank my family for all their love and encouragement in support of all my pursuits. Most of all, my loving, supportive, encouraging, and patient girlfriend Selina. The joy and enthusiasm she has for her research was contagious and motivational for me, especially during tough times in the Ph.D. pursuit.

Grenoble, January 4, 2017

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Abstract

Big Data highlights all the flaws of the conventional computing paradigm. Neuro-Inspired computing and other data-centric paradigms rather address *Big Data to as resources to progress*.

In this dissertation, we adopt Hierarchical Temporal Memory (HTM) principles and theory as neuroscientific references and we elaborate on how Bayesian Machine Learning (BML) leads apparently totally different Neuro-Inspired approaches to unify and meet our main objectives: (i) simplifying and enhancing BML algorithms and (ii) approaching Neuro-Inspired computing with an Ultra-Low-Power prospective. In this way, we aim to bring intelligence close to data sources and to popularize BML over strictly constrained electronics such as portable, wearable and implantable devices.

Nevertheless, BML algorithms demand for optimizations. In fact, their naïve HW implementation results neither effective nor feasible because of the required memory, computing power and overall complexity.

We propose a less complex on-line, distributed nonparametric algorithm and show better results with respect to the state-of-the-art solutions. In fact, we gain two orders of magnitude in complexity reduction with only algorithm level considerations and manipulations. A further order of magnitude in complexity reduction results through traditional HW optimization techniques. In particular, we conceive a proof-of-concept on a FPGA platform for real-time stream analytics.

Finally, we demonstrate we are able to summarize the ultimate findings in Machine Learning into a generally valid algorithm that can be implemented in HW and optimized for strictly constrained applications.

Keywords

Bayesian Machine Learning; Variational Bayesian Inference; Expectation-Maximization; On-line learning; Learning-On-a-Chip; Distributed Machine Learning; Message passing; Loopy Belief Propagation; Probabilistic Graphical Model; FPGA; Deep Learning; Neuromorphic Engineering; Probability Theory; Hierarchical Temporal Memory; Ultra-Low-Power; Optimizations.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Résumé

Les Big Data mettent en évidence tous les défauts du paradigme de l'informatique classique. Au contraire, le Neuro-Inspiré traite les Big Data comme ressources pour progresser.

Dans cette thèse, nous adoptons les principes de Hierarchical Temporal Memory (HTM) comme références neuroscientifiques et nous élaborons sur la façon dont le Bayesian Machine Learning (BML) mène les approches dans le Neuro-Inspiré à s'unifier et à atteindre nos objectives: (i) la simplification et l'amélioration des algorithmes de BML et (ii) l'approche au Neuro-Inspiré avec une prospective Ultra-Low-Power. Donc, nous nous efforçons d'apporter le traitement intelligent proche aux sources de données et de populariser le BML sur l'électronique strictement limitées tels que les appareils portables, mettable et implantables.

Cependant, les algorithmes de BML ont besoin d'être optimisés. En fait, leur mise en œuvre en HW est ni efficaces, ni réalisables en raison de la mémoire, la puissance de calcul requises.

Nous proposons un algorithme moins complexe, en ligne, distribué et non paramétrique et montrons de meilleurs résultats par rapport aux solutions de l'état de l'art. En fait, nous gagnons deux ordres de grandeur de réduction en complexité au niveau algorithmique et un autre ordre de grandeur grâce à des techniques traditionnelles d'optimisation HW. En particulier, nous concevons une preuve de concept sur une plateforme FPGA pour l'analyse en temps réel d'un flux de données.

Enfin, nous démontrons d'être en mesure de résumer les ultimes découvertes du domaine du BML sur un algorithme généralement valide qui peut être mis en œuvre en HW et optimisé pour des applications avec des ressources limitées.

Mots-clés

Apprentissage automatique bayésienne; Inférence variationelle bayésienne; Expectation-Maximization; Apprentissage en ligne; Apprendre-On-a-Chip; apprentissage automatique distribué; passage des messages; propagation de croyance en boucle; modèles graphiques probabilistes; FPGA; apprentissage en profondeur; ingénierie neuromorphique; théorie des probabilités; mémoire hiérarchique temporelle; très basse consumation énergétique; optimisations.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Contents

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Matteo Causo, Lille 1, 2017

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Matteo Causo, Lille 1, 2017

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Matteo Causo, Lille 1, 2017

List of Figures

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Matteo Causo, Lille 1, 2017

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Matteo Causo, Lille 1, 2017

List of Tables

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Chapter 1 Introduction

Nowadays, Big Data are undermining the mechanisms of technological progress that have worked so far by challenging the conventional computing paradigm: a shift to data-centric computing paradigms is crucial for progress to continue. Taking inspiration from the human brain seems an interesting solution, but the lack of understanding of the brain causes confusion and disagreement in the Neuro-Inspired field.

In this chapter, we elaborate on how Bayesian Machine Learning (BML) leads apparently totally different Neuro-Inspired approaches to unify and meet this PhD thesis main objectives.

1.1 Progress as a long same story

The technological progress our society has experienced in the last years is a direct consequence of the High-Tech Revolution, properly depicted in [Forester 1987]. According to the author, a similar explosion of technological and social innovation on such scale dates back a bit more than a century ago with the Second Industrial Revolution, i.e. the Technological Revolution. Automation of manufacturing processes led living standards to improve significantly as the prices of goods fell dramatically due to the increases in productivity [Wells 1889]. *Innovation brought new technology and technology allowed for further innovation*.

This mechanism is clearly implied in the well-known Moore's rule of thumb: since 1975, chips were designed and manufactured twice more complex than they had been just a year and a half before [Kish 2002]. However, according to [Forester 1987], the declining cost of computing power and memory is the first, but just one of the three major trends triggering the High-Tech Revolution. Together with it, the adoption of a universal machine language, i.e. the binary code, and the deregulation and privatization in the telecommunication domain: a multidisciplinary effort of electronics, computer science and telecommunications in order to store, manipulate and transmit the information, referred to as Information and Communication Technology (ICT) [Zuppo 2012].

In ICT, the focus has then moved on data, which are the manageable representation of information. In particular, the more the technological progress, the more the data produced then, the more the resources required. Whether necessity is the mother of innovation, the same mechanism previously identified for the Technological Revolution applies. In other words, *data foment the technological progress by challenging innovation*.

1.2 Progress in the Big Data era

With the end or pseudo-end of the Moore's law [Waldrop 2016; Kish 2002], this well-oiled self-contained mechanism for progress destabilizes. Device scaling increases device variability and lowers Signal-to-Noise Ratio (SNR). Furthermore, the huge amount of data, the High-Tech society is constantly producing, is highlighting all the flaws of the conventional computing paradigm that has worked so far mainly due to the inertia generated by the technological progress. In the Big Data era [McAfee and Brynjolfsson 2012], big slower memories inflate the well-known Von Neumann's bottleneck. Processors clock frequencies, limited by power budget and heat dissipation, encourage for multi-cores architectures that lead to parallel computing [Asanovic et al. 2006].

All these problems decree the end of the conventional computing paradigm and identify the requirements of the next one, for the progress to continue. In particular, fully *distributed* concurrent processing and memory is required to simplify and focus computing on local data. In-memory computing and *on-line*, i.e. on-the-fly, elaboration of new incoming data becomes a paramount requirement in order to address such colossal amount of data. *Power consumption* has to be strictly constrained and *tolerance* to fault and components variability intrinsically guaranteed.

A first indication on the direction to choose is that of data-centric computing. Data are no longer ends in themselves or in the processing they were collected for, but they become a source of new information and knowledge. For example, to date, photos were mainly used as memories, but with a huge amount of them, the goal shifts to extract insight on the behavior and lifestyle of an individual for a variety of reasons such as to offer customized business solutions, national security or to study social trends.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

The well-known statistical regression analysis in Figure 1:1 gives an intuition on how this is possible and why Big Data is key. The goal is to identify the phenomenon implied by the data that is generally unknown, i.e. hidden, and then to infer the mathematical function that links the data and parameters of the observed system. The more data collected, the easier the identification of the hidden phenomenon with results simple and accurate. Nevertheless, Big Data are generally collected from different sources and often they present mixed types or have no predefined models, also known as unstructured Big Data. Once again, the need for distributed architectures that benefit of local data similarities and regulate system-level information fusion becomes evident.

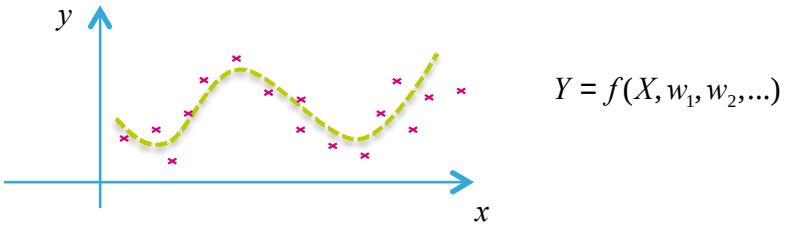


Figure 1:1 Statistical regression analysis.

(The more the data (X, Y), the easier to retrieve the hidden phenomenon $f(\cdot)$ that may be dependent on some parameters $w_{i\in\mathbb{N}^+}$)

Today's market and technological trends push towards the Internet of Things (IoT) and a more and more intimate and natural human-machine integration, which require overcoming conventional computing paradigm limitations.

1.3 All signs point to the human brain

The human brain is an amazingly complex and efficient machine. It performs a set of functions such as feature extraction, classification, synthesis, recognition, learning, and high-order decision-making admirably well. Neural processing properties present an attractive alternative to the conventional computing paradigm in deeply scaled regimes: it thrives on randomness and variability, and massive parallelism, major redundancy and adaptability are of essence.

The human brain presents robustness against component failure and variations, amazing performance with mediocre components, seamless interaction with the analog world and up to 2-3 orders more efficiency than today's silicon equivalent, i.e. its average power consumption has been estimated at about 20 W [Drubach 2000; Hart 1975; Chowdhary et al. 2014]. Furthermore, memory and logic are atomically interleaved and data are elaborated on-line [Di Ventra et al. 2009; Di Ventra and Pershin 2013; Koch 2004].

Taking inspiration from the human brain is, therefore, a winning rather fashionable strategy as reflected the huge excitement and enthusiasm unleashed around the so-called Neuro-Inspired computing paradigm. Recently, public and private consortia, as well as governments, have allocated major funding to study the human brain and to appropriate its main features. Then, exponential progress resulted in this field along with so much confusion and disagreement on theories, principles, common objectives and future trends and developments. In fact, *we do not yet know how the human brain works, there is no dominant design and every solution stays equally valid, providing consistency in its results*[1].

1.3.1 Evolution and current trends in Neuro-Inspired computing

There are two contrasting approaches in the Neuro-Inspired field. The first is the canonical scientific approach engineering widely exploits: *simplification, modeling and replication*. Since neurons were functionally recognized by Santiago Ramón y Cajal [Llinás 2003], several efforts focused on correctly modeling Neural Networks (NNs) at functional and circuit levels [Cowan 1990]. In 1957, Frank Rosenblatt invented the perceptron algorithm, i.e. a single-layer Artificial NN (ANN) capable of linear multiclass classification, by exploiting the McCulloch-Pitts artificial neuron in Figure 1:2 with binary activation function g. Twenty-five years later, in 1982 Paul Werbos employed the back-propagation algorithm for training a Multi-Layer Perceptron (MLP) with a smooth and differentiable g, i.e. a multi-layer ANN for non-linear multiclass classification. This approach limps in 2000s, because other intelligent computing methods such as Support Vector Machine (SVM) showed superior performance to ANN [Schmidhuber 2015].

[1] This represents my personal takeaway summarizing the "Thursday discussion session" of the 2016 Neuro-Inspired Computational Elements (NICE) Workshop hold at Hellen Wills Neuroscience Institute - UC Berkeley on March 7-9, 2016. The most acknowledged experts from different fields argued that further fundamental studies must be carried on and reference benchmarks must be identified for a dominant design to appear, because we do not yet know how the human brain works.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

As of 2009, the availability of huge computational power finally allowed for simulating Deep NNs (DNNs) [Raina et al. 2009], i.e. ANNs with many hidden layers, and set the stage for the second approach, demonstrating that *inherent complexity and non-linearity in the human brain structure must be preserved* [Mallat 2016]. Therefore, the focus shifted towards identifying engineering mechanisms to deal with complexity and solve the gradient vanishing problem of back-propagation [Schmidhuber 2015]. For instance, in 2011 Xavier Glorot introduced a more biologically plausible g, the rectifier [Glorot et al. 2011], which improved DNNs classification accuracy. However, today's best-in-class accuracy is achieved by Deep Learning (DL) models and algorithms that create a hierarchy of invariant features [Le et al. 2011] in the first levels of DNN, so that final classification simplifies and improves. Nevertheless, DL has three hard problems: it consumes tremendously in terms of learning time, power and resources; invariance of features is built at the cost of seeing huge amounts of data and no manipulation or understanding is required for hidden layers. In fact, they remain black boxes, which design follows the "*bigger is better*" rule of thumb[2], with no room for optimizations with respect to the low-power objective.

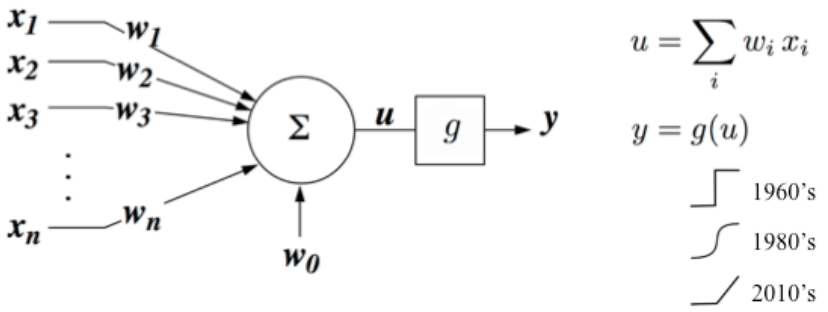


Figure 1:2 A brief history of Neural Networks (NN).

("Beyond inspiration: Three lessons from biology on building intelligent machine" at NICE 2016, of Bruno Olshausen)

In 1989, Carver Mead noted the analogy between charges moving in MOS transistors operated in weak inversion and charges flowing across the membranes of neurons. He claimed the possibility to *mimic neurobiological architectures* present in the nervous system with Very-Large Scale Integration (VLSI) electronic components in analog, digital, and mixed-mode circuits [Mead and Ismail 1989]. Upon this idea, the *Neuromorphic Engineering* approach established. For instance, the mechanisms of human hearing and eyesight have been successfully recreated in the artificial cochlea and retina [Liu and Delbruck 2010]. MOS circuits for neurons and synapses resulted from biological and neuroscientific studies and led to Spiking NNs (SNN) [Maass 1997]. Consequently, the theory for on-line local unsupervised learning was developed through temporal correlation of subsequent spikes[3], i.e. the Spike-Timing Dependent Plasticity (STDP) [Markram et al. 1997]. Neuromorphic Engineering became an university course[4] and first deeply-rooted-in-biology HW capable of on-line learning appeared [Indiveri et al. 2016]. Furthermore, nanotechnology recently raised further interest with *memristors* [Strukov et al. 2008] and the demonstration of their synaptic behavior [Kuzum et al. 2012; La Barbera et al. 2015]. Thus, heterogeneous integration with MOS technology is highly envisaged for massively parallel and dense HW with interleaved memory and logic. However, with the objective of mimicking the human brain with a limited power and resources budget, the Neuromorphic Engineering approach intrinsically proceeds bottom-up, without an overall application-driven view.

Albeit different, DL and Neuromorphic Engineering exhibit the same basic problem: profound principles of brain functioning are not yet known, therefore *conscious* or *conscientious* network design is not yet possible. Because of this nuanced indefiniteness at the basis of Neuro-Inspired, there is not yet a dominant design and many hybrid approaches appeared. For example, Neuromorphic Computing [Esser et al. 2016; Hof 2014] focuses on dedicated HW solutions devoted to application-driven high-performance computing, namely IBM TrueNorth [Diehl et al. 2016] and SpiNNaker [Knight et al. 2016]. Nevertheless, they suffer the same limitations of both DL and Neuromorphic Engineering.

Since 2013, understanding how the human brain works has been a motivation for both the European Union Human Brain Project (HBP) and the Obama's BRAIN Initiative: inspiration drawn from the human brain definitely allows progress to address Big Data challenges. At the same time, *through the human brain emulation, useful behaviors are gathered to better identify its working principles and improve emulation itself*: the usual mechanism for progress is primed, but in a new formulation.

[2] Andrew Ng, *Deep Learning - CS 229: Machine Learning*, Lecture Notes, Stanford University

[3] Voltage spikes model characteristic electrical pulses called Action Potentials (AP), which are generated by neurons and rapidly propagated over large distances by nerve fibers. A sequence of spikes, referred as spike train, encodes information based on different coding schemes studied in neuroscience as Neural Coding [Brown et al. 2004].

[4] University of Zurich and ETH Zurich Neuromorphic Engineering courses in the frame of the MSc program on Neural Systems and Computation.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Matteo Causo, Lille 1, 2017

1.3.2 Bayesian Machine Learning as common ground

Ability is the true marker of understanding according to Richard Feynman's quote, popular in featuring progress in Neuro-Inspired field, i.e. "What I cannot create, I do not understand". This sentence actually culminated with a less-known conclusion: "Know how to solve every problem that has been solved". Therefore, Feynman meant that starting with a blank piece of paper and the knowledge already in his mind, any theoretical result he could re-derive, he understood.[5] This is exactly the intuition developed and supported in the present PhD thesis to overcome today's Neuro-Inspired limitations: a generally valid approach is required in the Neuro-Inspired field in order to interface and unify different strategies; draw a logical thread, a manageable model, common mathematical basis and finally make order among them. In-depth knowledge and comprehensive mastery of this approach together with acknowledged roots in neuroscience will, then, provide the tool for opportunely emulating the human brain.

Very promising is the Bayesian approach to Machine Learning, referred to as Bayesian Machine Learning (BML) that is a probabilistic and statistical approach, thus considered in both Neuro- and Shannon-Inspired computing paradigms [Shanbhag et al. 2008].

Recent and encouraging developments consistently link DL [Gal and Ghahramani 2015b; Gal and Ghahramani 2015a; Ghahramani 2015] and learning and inferring mechanisms in SNN [Nessler et al. 2013; Deneve 2015; Pecevski and Maass 2016] to BML, so that convergence to it becomes increasingly evident. Although these are very important results, their employment mirrors every Neuro-Inspired approach discussed so far. In other words, BML is used to explain and validate NNs behavior but known limitations remain.

In this PhD thesis work, we reverse the way of applying and thinking of BML. In particular, we employ BML to accurately define the resources and the system to build. In this way, we will have a punctual control of the neuro-inspired HW and a-priori understanding of each element function. Therefore, our approach to Neuro-Inspired is *top-down* and makes *conscious* and *conscientious* optimizations feasible.

1.3.3 Bayesian Machine Learning and Neuroscience

Since 1860s with Hermann Helmholtz's work, experimental psychology had been explaining human brain's cognitive abilities in terms of Bayesian probabilistic models [Jaynes 1988; Rao and Ballard 1999; Lee and Mumford 2003; Gold and Shadlen 2007; Westheimer 2008]. In fact, the nervous system inherently operates with uncertainty and organize sensory data into an accurate internal model of the outside world [Fahlman et al. 1983]. Therefore, insights into nervous system operating principles could be drawn by Bayesian cognitive science [Griffiths and Tenenbaum 2006] though BML [Ghahramani 2004]. Furthermore, the theoretical settings of hierarchical Bayesian inference has gained acceptance in neuroscience as a possible framework for understanding the computational and information processing roles of cortical circuits in the human brain [Hegdé and Felleman 2007; Doya et al. 2006; Deneve 2015; Lee and Mumford 2003]. However, not all neuroscientists share this same belief and a relentless ongoing debate started to question it.[6]

We dissociate from this discussion and endorse Konrad Paul Kording's opinion on this subject [Kording 2014]: "The Bayesian framework is a probabilistic tool, generally applicable and crucial for analyzing uncertainty in complex data we have no clues about. Analyzing neural data with Bayesian statistics is efficient and gives better results with less training data. Assuming the brain solves problems close to the Bayesian optimum often predicts behavior and finally, asking how the nervous system could achieve Bayesian behavior leads to new insights about neural codes".

1.3.4 A convincing theory of reference

Particular interest into the Neuro-Inspired field recently grown on Hierarchical Temporal Memory (HTM) that is a biologically "constrained" theory of neocortex, derived from neuroanatomy and neurophysiology, which aims to explain how neocortex and a few related brain structures work [Hawkins et al. 2016]. At first, the distinction between biological emulation, theoretical formulation and implementation was not clear. In fact, in [George and Hawkins 2009], HTM was defined as the Bayesian formulation of the Memory-Prediction Framework proposed in [Hawkins 2005] with the explicit goal to map the mathematics of HTM to cortical-thalamic and the microcircuits of cortical columns.

In a second revision, HTM became a Machine Learning technology [Hawkins 2011] and the focus shifted on its SW implementation through Cortical Learning Algorithms (CLA). Recently, attention came back on properly distinguishing between biologically compliant

[5] From Mark Eichenlaub's answer to What did Richard Feynman mean when he said, "What I cannot create, I do not understand"?
[6] On December 4-5, 2015 the NYU *Center for Mind, Brain, and Consciousness* hosted a conference on "Is the Brain Bayesian?".

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Matteo Causo, Lille 1, 2017

HTM principles and theory, and their actual implementation. In fact in [Hawkins et al. 2016], the author points out that their "goal isn't to recreate a brain, but to understand how brains work in sufficient detail so that [they] can test the theory biologically and also build systems that, although not identical to brains, work on the same principles. Sometime in the future designers of intelligent machines may not care about brains and the details of how brains implement the principles of intelligence. The field of machine intelligence may by then be so advanced that it has departed from its biological origin."

In the present PhD thesis work, we adopt HTM principles and theory as neuroscientific references. Therefore, following is a summary extracted from [Hawkins 2011; Hawkins et al. 2016].

The human brain consists of several components successively added through evolution to incrementally improve its behaviors. In fact, some very basic components have equivalent counterparts in simple animals. Typically, new components have been added on top of old ones for a high-level control of them. The neocortex is the most recent addition to mammals' brain and it dominates the human brain, occupying about 75% of its volume [Noback et al. 2005]. Thus, there is common agreement in thinking that it is the seat of intelligence.

Hierarchy. Because of this evolutionary path, every brain is physically and logically a hierarchy of components defined by connectivity. The actual hierarchy is complicated because connections often skip levels and go sideways between parallel paths. Anyhow, the hierarchical structure of the neocortex is well established [Zingg et al. 2014]. HTM represents regions as levels in the hierarchy. Convergence is ensured by ascending the hierarchy as multiple elements in a child region converge onto an element in a parent region. However, due to feedback connections, divergence occurs by descending the hierarchy. As for DL, sharing representations in a hierarchy leads to generalization of expected behavior and enables new concepts in the world to inherit the known properties of their subcomponents. A simple but explanatory experiment[7] involves looking at Figure 1:3. At a first glance, it results really difficult to figure out what the image is about. In fact, by ascending the hierarchy neither concept we have in mind seems to be revealed. So, we cannot generalize the strange shapes we see and try to recognize and connect. Instead, if somebody told us there is a cow in Figure 1:3, the reverse mechanism would be triggered and by descending the hierarchy, we would try to trace the contours and identify the image piece by piece according to what we would be looking for, i.e. the concept of the cow.

Figure 1:3 Mysterious image.

(What Can You See? – Some Questions About the Content of Visual Experience, of T. McClelland)

Homogeneity. The most remarkable aspect of the neocortex is its homogeneity, in fact, it displays a uniform pattern of neural circuitry as its types of cells and their connectivity are nearly identical with relatively minor variations. Biological evidences suggest that the human neocortex got large by replicating a basic element over and over, therefore every part of the neocortex must be performing many different cognitive/intelligent functions with the same neural algorithms. HTM theory focuses on understanding these common algorithms.

Sparse Distributed Representation (SDR). Through mutual inhibition among neurons, information in the brain is always *sparsely* represented by a small percentage of active neurons, although they are highly interconnected. However, simultaneous *distributed* activations of specific neurons are required to encode representation semantics. In fact, two representations that activate the same neuron, share a semantic property and vice versa a set of neurons can

[7] Bruno Olshausen at Neuro-Inspired Computational Elements (NICE) Workshop on March 7, 2016.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Matteo Causo, Lille 1, 2017

simultaneously activate multiple representations without confusion. Thus, in SDR the representation and its meaning are atomically related [Kanerva 1992].

Sensory Encoders. Sensory organs interface the world with the brain by preprocessing the input information that becomes digestible to the brain. For example, in the human visual system the retina preprocesses the information gathered by the photoreceptor cells and sends this information to the brain via the optic nerve. Basically, the retina processes an image from the focused light, and the brain is left to decide what the image is. Sensory organs are referred as *encoders* in the HTM terminology and are specific to particular data type.

Closed-Loop system. The neocortex is not a standalone system, but it is part of a closed-loop system, where it learns how to interact with and control the older brain components to create novel and improved behaviors. A region of neocortex works in a purely statistical realm: it doesn't "know" what its inputs represent, what its output might do or where it is placed in the hierarchy. It is told what behaviors the older brain components are generating as well as what they are sensing. Therefore, the sensory-motor integration is a nearly universal property of the brain: it is fundamental for learning the structure of the world in the neocortex and for realizing the full potential of the HTM theory.

Sequential Memory. The brain learns different patterns underlying the same concept by observing them in sequence. In other words, time supervises learning by teaching which spatial patterns go together. HTM is based on a Temporal Memory that is a memory of sequences and in order to learn, it must be exposed to time-changing inputs.

On-line learning, inferring, predicting and attention. The human brain learns continuously and in an on-line, i.e. on-the-fly fashion by building a predictive model of the world, for which it is predicting what it expects will happen next. The prediction is compared to what actually happens and forms the basis of continuous learning. Therefore, prediction is continuous; occurs at every level of the hierarchy; is context-sensitive; tells if new inputs are expected or not; allows for attention and helps to make the system more robust to noise. In fact, by minimizing the predictions error, learning becomes sort of self-supervised despite it is globally unsupervised. At the same time, learning and recognizing sequences are the basis of forming predictions. Like a biological system, when an HTM predicts what is likely to happen next, the prediction can focus the attention or bias the system toward inferring what it predicted, like for the experiment of Figure 1:3. Successfully matching new inputs to previously stored sequences is the essence of inference. There is no need for a learning phase separate from an inference phase, though inference improves after additional learning. Another advantage of continuous learning is that the system will constantly adapt if the patterns in the world change. HTM region must handle novel input during inference and training.

We depart from the proposed implementation of HTM we deem as Neuromorphic Computing. In fact, CLA are memory-based and demand huge memory capacity and long data presentation [Hawkins 2011]. Furthermore, Neuro-Inspired limitations apply and a HW implementation of CLA [Deshpande 2011; Zhou and Luo 2013; Zyarah 2015] would have fallen into the Neuro-Inspired mainstream, thus in accord with neither the intuition behind this PhD thesis nor its key objectives introduced below.

1.4 This PhD thesis objectives

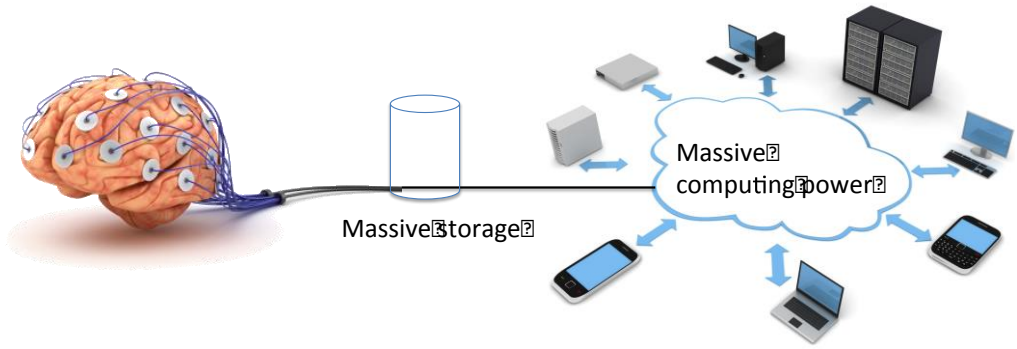
Our paramount objective is to approach Neuro-Inspired computing with an Ultra-Low-Power (ULP) prospective, so that to bring intelligent processing as close as possible to data sources and to popularize Machine Learning. In fact, IoT and medical applications, such as the Human Intranet [Rabaey 2015], Google Smart Lens, Siri and Microsoft HoloLens can widely benefit of low cost, small, low-power, intelligent signal processing systems.

An illustrative example of target application is the epileptic seizure detection and prediction on electroencephalogram (EEG) recordings. An epileptic seizure is a brief episode of signs or symptoms due to abnormal excessive or synchronous neuronal activity in the brain [Fisher et al. 2005]. The outward effect can vary from uncontrolled jerking movement to a momentary loss of awareness. State-of-the-art solutions include surgery for removing the part of the brain imputed to trigger the seizure with all the risks this may entail or warning systems, which do not prevent seizures, but only allow the patient to have the time to prepare for it. In this case, a driver may pull over or a swimmer can leave the water and their life expectation augments. The ultimate goal of these systems is actually to predict the seizure even a few seconds before in order to put in place counterspiking or local drugs release in a closed-loop system to completely block the seizure [Ramgopal et al. 2014]. Therefore, epileptic seizure detection and prediction address unstructured Big Data; rely on mediocre equipment with poor spatial resolution and very low Signal-to-Noise Ratio (SNR); entail bio-signals processing and distributed sensing; offer challenges for Ultra-Low-Power implantable electronics; require continuous

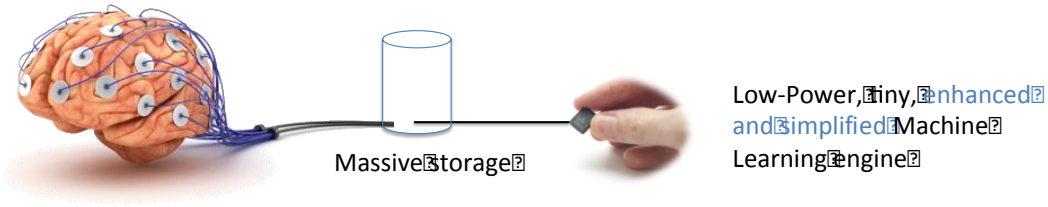
© 2017 Tous droits réservés.

lilliad.univ-lille.fr

intelligent processing as seizures are deterministically unpredictable and vary from patient to patient and from time to time; allow for distributed processing and for plenty of possible improvements. In fact, the state-of-the-art solution depicted in Figure 1:4a requires off-line, ex-situ, power-hungry Machine Learning algorithms to process pre-stored EEG recordings. It is neither low-power nor implantable, thus not suitable for continuous monitoring and preventive actuation.

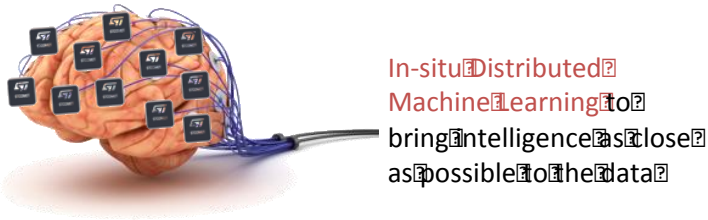


(a) State-of-the-art solution for epileptic seizure warning systems.



(b) Small ASIC with Learning-on-a-Chip capability: in-situ ULP processing.

(c) On-line learning of continuous streams: no longer need for massive storage.



(d) In-sensing intelligent processing distributed on each EEG electrode.

Figure 1:4 This PhD thesis objective explained on successive illustrations.

Our objective can be reformulated as simplifying and enhancing Machine Learning algorithms by taking advantage of the most innovative theories in the Neuro-Inspired field; designing the ULP small Machine Learning engine in Figure 1:4b with Learning-on-a-Chip capabilities for in-situ intelligent processing; disposing of massive storage for long EEG recordings by on-line elaboration of the continuous streams of data, Figure 1:4c, and finally distributing the intelligent processing among several collaborative nodes, Figure 1:4d, in order to work on small amount of local data and to push elaboration as close as possible to data sources, i.e. EEG electrodes or sensors.

We aim to build a Neuro-Inspired ULP HW that closely meets HTM principles and theory. It has to be general purpose, capable of on-line learning, continuous inferring and predicting. It has to start without any knowledge on its inputs and the elaboration it has to perform, rather it learns everything from data and continuously adapts its internal representation of the external world, if it is

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

evolving and non-stationary. Complexity and hierarchy have to be automatically and naturally incorporated by a top-down application-driven design, which leads to *conscious* and *contentious* optimizations.

1.5 Main idea behind this PhD thesis

The main objective of this PhD thesis seems too ambitious and pretentious in light of the known Neuro-Inspired limitations. In fact, realizing an ULP HW out of a Machine Learning algorithm that employs Cloud Computing for days in processing colossal amounts of data sounds like science fiction; conceiving a HW compliant with overall HTM theory and principles results challenging and reversing Neuromorphic Engineering approach from bottom-up to application-driven top-down dumbfounds, but it is the only way to get an idea on how to design the HW with full knowledge of components, their connections and the leeway to optimization.

The human brain automatically manages its complexity, optimizes its structure and consistently reorganizes its hierarchy of concepts. Fundamental principles for the brain optimal design are therefore already present in nature, but not in Neuro-Inspired simplistic models. Neuroscience studies the ability of plastic reorganization of our brain referred to as neuroplasticity[8] and identifies several degrees-of-freedom the human brain relies on. In fact, what is generally modeled in Neuro-Inspired field is *functional* plasticity, i.e. the strength of existing synaptic connections change [Attneave et al. 1950]. Human brain is however also capable of *structural* plasticity, that is morphological and anatomical changes correlated with learning and experiencing[9]. Thus, new dendritic spines are born constantly and consequently new synapses appear. Conversely, dendritic spines and synapses can disappear or shift in other positions so that to change their contribution to neurons spiking events. Even in adult age, new neurons grow [Altman 1962; Arvidsson et al. 2002] and new functional neural networks are formed. In addition, the brain is constantly adapting to accommodate new information, but during sleep it optimizes to compact memories and reorganize its structure. In other words, our brain processes and encodes the information by defining and populating its structure in a complex and highly plastic way: new data are processed on-the-fly and contribute to shape the brain structure itself. Figure 1:5 shows an illustration of these principles.

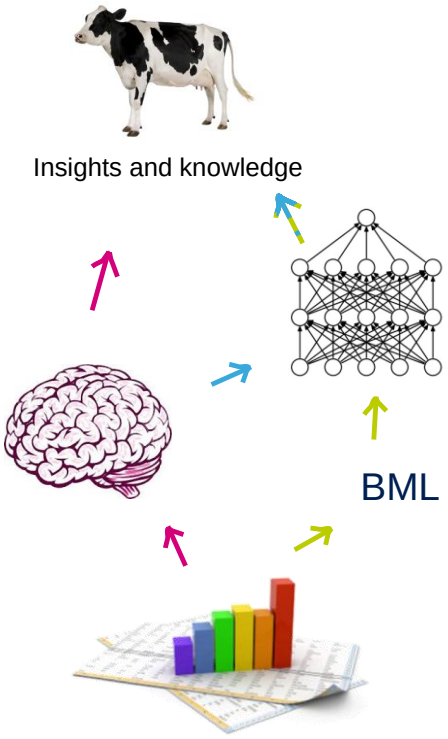


Figure 1:5 Simplistic illustration of our idea.

(The human brain retrieves insights and knowledge from raw or preprocessed inputs (fuchsia arrows). It however employs its plastic structure to process information and model the world (light-blue arrows). Our idea is to exploit Bayesian Machine Learning (BML) to model the world and to retrieve insights and knowledge from input data (light-green arrows).)

[8] What Is Brain Plasticity?, By Kendra Cherry, June 14, 2016
[9] Synapses, Neurons and Brains, Idan Segev, Hebrew University of Jerusalem, Coursera 2016

© 2017 Tous droits réservés.

Thèse de Matteo Causo, Lille 1, 2017

No HW implementation can evolve over time, so it is unable of this intrinsic plasticity. A SW rather could, but the human brain's fundamental principles are not yet known, thus the human brain's functioning results complex and the complexity is solved as *bigger is better*. This explains the origin of the Neuro-Inspired limitations respectively in Neuromorphic Engineering and DL.

BML instead, broadly tracks the behavior of the brain because it is capable of plastically building a hierarchical and complex model of the world through the processing of input data. This model, referred to as a Probabilistic Graphical Model (PGM) is implicitly created and modified in the probabilistic space and therefore it is not real, Figure 1:5. The idea developed in this PhD thesis is that we can exploit this exceptional ability of BML and implement a HW that does not change in the real world, but the projection of its elaboration in the probabilistic world is dynamic, plastic and manages the complexity implicitly. Furthermore, the BML covers and manipulates inference, prediction, on-line learning and includes tools for sequential data analysis as required in the HTM theory.

Certainly, the BML does not use the sophisticated techniques of the brain to shape its structure: it selects the most suitable model among infinite ones rather than constructively building the model from data. However, this is in line with Kording's opinion. In fact, the Bayesian approach does not necessary encode the real behavior of the brain, instead it is a probability model that thanks to the Law of Large Numbers well explains it. In this regard, we are convinced that the Bayesian approach gathers in their entirety the brain's mechanisms fragmentarily and differently exploited by each of the Neuro-Inspired models. Therefore, our idea is to reinterpret recent findings that link Neuro-Inspired approaches to BML [George and Hawkins 2009; Deneve 2015; Gal and Ghahramani 2015b] with the new perspective of considering Neuro-Inspired approaches as particular implementations of the BML, so that to theoretically derive the guidelines for conscious and contentious optimizations.

With this in mind, we can focus on conveniently defining a BML algorithm to meet this PhD thesis objectives independently on its actual implementation, i.e. SNN, DL accelerator, HTM, Bayesian Processor or even breaking out of the neural patterns with automata processors.

1.6 Conclusions

Big Data is highlighting all the flaws of the conventional computing paradigm that has worked so far mainly due to the inertia generated by the technological progress. Conversely, data-centric paradigms address *Big Data to as resources to progress*. Consequently huge excitement and enthusiasm unleashed around the so-called Neuro-Inspired computing paradigm. Then, exponential progress resulted in this field along with so much confusion and disagreement. In fact, profound principles of brain functioning are not yet known, thus there is not yet a dominant design but many hybrid approaches.

In the present PhD thesis work, we adopt HTM principles and theory as neuroscientific references and we elaborate on how Bayesian Machine Learning (BML) leads apparently totally different Neuro-Inspired approaches to unify and meet this PhD thesis main objectives. In particular, our idea is to carry out the brain working principles gathered in the HTM theory, through BML and its exceptional ability to create and plastically modify an implicit hierarchical model of the observed world. Then, we reverse the way we apply and think of BML and result with a *top-down* approach that allows for *conscious* and *conscientious* optimizations.

In Chapter 2, the BML framework is introduced so as to provide a background and a common mathematical convention. The idea behind this PhD thesis is more precisely identified together with its theoretical roots and feasibility. Chapter 3 is devoted to analyze the state-of-the-art VBEM HMM GMM complexity and to propose algorithm level optimizations with the objective to make its HW implementation both feasible and attractive. The proposed distributed on-line VBEM HMM GMM is derived, analyzed and validated by means of both synthetic data and real-world applications. In Chapter 4, a proof-of-concept HW implementation in conventional digital CMOS electronics is derived and tested on FPGA platform. Interesting studies related to this PhD thesis subject are encountered in Chapter 5, where convergence and performance of a BML algorithm of interest are improved though inference; a novel ultra-low-power Feature Extraction Engine (FEE) for biosignals is proposed and three related internship works, the author tutored, are summarized. Conclusions are finally drawn in Chapter 6 together with future works.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Matteo Causo, Lille 1, 2017

References

Joseph Altman. 1962. Are New Neurons Formed in the Brains of Adult Mammals. *Science (80-.).* Volume, 3509 (1962), 1127–1128. DOI:http://dx.doi.org/10.1126/science.135.3509.1127

Andreas Arvidsson, Tove Collin, Deniz Kirik, Zaal Kokaia, and Olle Lindvall. 2002. Neuronal replacement from endogenous precursors in the adult brain after stroke. *Nat. Med.* 8, 9 (2002), 963–970. DOI:http://dx.doi.org/10.1038/nm747

Krste Asanovic, Bryan Christopher Catanzaro, David a Patterson, and Katherine a Yelick. 2006. The Landscape of Parallel Computing Research : A View from Berkeley. *EECS Dep. Univ. Calif. Berkeley Tech Rep UCBEECS2006183* 18 (2006), 19. DOI:http://dx.doi.org/10.1145/1562764.1562783

Fred Attneave, M. B., and Do O. Hebb. 1950. The Organization of Behavior; A Neuropsychological Theory. *Am. J. Psychol.* 63, 4 (October 1950), 633. DOI:http://dx.doi.org/10.2307/1418888

Selina La Barbera, Dominique Vuillaume, and Fabien Alibart. 2015. Filamentary switching: Synaptic plasticity through device volatility. *ACS Nano* 9, 1 (2015), 941–949. DOI:http://dx.doi.org/10.1021/nn506735m

Emery N. Brown, Robert E. Kass, and Partha P. Mitra. 2004. Multiple neural spike train data analysis: state-of-the-art and future challenges. *Nat. Neurosci.* 7, 5 (2004), 456–61. DOI:http://dx.doi.org/10.1038/nn1228

Kuntal Chowdhary, Jingyan Wang, Saavan Patel, and Shruti Koti. 2014. An Interview with Professor Jan Rabaey: Neural Prosthetics and Their Future Applications. *Berkeley Sci. J.* 18, 2 (2014).

Jack D. Cowan. 1990. Discussion: McCulloch-Pitts and related neural nets from 1943 to 1989. *Bull. Math. Biol.* 52, 1–2 (1990), 73–97. DOI:http://dx.doi.org/10.1007/BF02459569

Sophie Deneve. 2015. Bayesian Inference with Spiking Neurons. In *Encyclopedia of Computational Neuroscience*. 361–364. DOI:http://dx.doi.org/10.1007/978-1-4614-6675-8_568

Mandar Deshpande. 2011. FPGA Implementation & Acceleration of Building blocks for Biologically Inspired Computational Models. *Training* (2011).

Peter U. Diehl, Bruno U. Pedroni, Andrew Cassidy, Paul Merolla, Emre Neftci, and Guido Zarrella. 2016. Truehappiness: Neuromorphic emotion recognition on truenorth. *arXiv Prepr. arXiv1601.04183* (2016).

K. Doya, S. Ishii, A. Pouget, and R.P.N. Rao. 2006. *Bayesian Brain* Kenji Doya, Shin Ishii, Alexandre Pouget, & Rajesh P. N. Rao, eds., The MIT Press.

Daniel Drubach. 2000. *The brain explained*, Prentice Hall.

Steven K. Esser et al. 2016. Convolutional networks for fast, energy-efficient neuromorphic computing. *Proc. Natl. Acad. Sci.* 113, 41 (October 2016), 11441–11446. DOI:http://dx.doi.org/10.1073/pnas.1604850113

Scott E. Fahlman, Geoffrey E. Hinton, and Terrence J. Sejnowski. 1983. Massively Parallel Architectures for Al : NETL, Thistle, and Boltzmann Machines. *Aaai-83* , June (1983), 109–113.

Robert S. Fisher et al. 2005. Epileptic seizures and epilepsy: Definitions proposed by the International League Against Epilepsy (ILAE) and the International Bureau for Epilepsy (IBE). *Epilepsia* 46, 4 (2005), 470–472. DOI:http://dx.doi.org/10.1111/j.0013-9580.2005.66104.x

Tom Forester. 1987. *High-Tech Society: The Story of the Information Technology Revolution* Basil Blackwell, ed., Oxford: MIT Press.

Yarin Gal and Zoubin Ghahramani. 2015a. Bayesian Convolutional Neural Networks with Bernoulli Approximate Variational Inference. (2015), 12. DOI:http://dx.doi.org/10.1029/2003JD002581

Yarin Gal and Zoubin Ghahramani. 2015b. On Modern Deep Learning and Variational Inference. *Adv. Approx. Bayesian Inference Work. NIPS* (2015), 1–9.

Dileep George and Jeff Hawkins. 2009. Towards a mathematical theory of cortical micro-circuits. *PLoS Comput. Biol.* 5, 10 (October

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Matteo Causo, Lille 1, 2017

2009), e1000532. DOI:http://dx.doi.org/10.1371/journal.pcbi.1000532

Zoubin Ghahramani. 2015. Probabilistic machine learning and artificial intelligence. *Nature* 521, 7553 (2015), 452–459. DOI:http://dx.doi.org/10.1038/nature14541

Zoubin Ghahramani. 2004. Unsupervised Learning. In *Machine Learning*. 6727–80. DOI:http://dx.doi.org/10.1007/BF00993379

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep Sparse Rectifier Neural Networks. *Aistats* 15 (2011), 315–323. DOI:http://dx.doi.org/10.1.1.208.6449

Joshua I. Gold and Michael N. Shadlen. 2007. The neural basis of decision making. *Annu. Rev. Neurosci.* 30 (2007), 535–574. DOI:http://dx.doi.org/10.1146/annurev.neuro.29.051605.113038

Thomas L. Griffiths and Joshua B. Tenenbaum. 2006. Optimal predictions in everyday cognition. *Psychol. Sci.* 17, 9 (2006), 767–773. DOI:http://dx.doi.org/10.1111/j.1467-9280.2006.01780.x

L.A. Hart. 1975. *How the Brain Works: A New Understanding of Human Learning, Emotion, and Thinking*, Basic Book.

J. Hawkins. 2005. On Intelligence. *Book* (2005), 1–174.

J. Hawkins, S. Ahmad, S. Purdy, and A. Lavin. 2016. *Biological and Machine Intelligence (BAMI)*,

Jeff Hawkins. 2011. HTM Cortical Learning Algorithms. (2011), 1–68.

Jay Hegdé and Daniel J. Felleman. 2007. Reappraising the functional implications of the primate visual anatomical hierarchy. *Neuroscientist* 13, 5 (2007), 416–421. DOI:http://dx.doi.org/10.1177/1073858407305201

Robert D. Hof. 2014. Neuromorphic Chips. *MIT Technol. Rev.* 117, 3 (2014), 54–57. DOI:http://dx.doi.org/10.1038/scientificamerican0505-56

Giacomo Indiveri, Federico Corradi, and Ning Qiao. 2016. Neuromorphic architectures for spiking deep neural networks. In *Technical Digest - International Electron Devices Meeting, IEDM*. 4.2.1-4.2.4. DOI:http://dx.doi.org/10.1109/IEDM.2015.7409623

Edward T. Jaynes. 1988. How does the brain do plausible reasoning? In *Maximum-entropy and Bayesian methods in science and engineering*. Springer, 1–24.

Pentti Kanerva. 1992. *Sparse Distributed Memory A study of psychologically driven storage*,

Laszlo B. Kish. 2002. End of Moore's law: Thermal (noise) death of integration in micro and nano electronics. *Phys. Lett. Sect. A Gen. At. Solid State Phys.* 305, 3–4 (2002), 144–149. DOI:http://dx.doi.org/10.1016/S0375-9601(02)01365-8

James C. Knight, Philip J. Tully, Bernhard A. Kaplan, Anders Lansner, and Steve B. Furber. 2016. Large-Scale Simulations of Plastic Neural Networks on Neuromorphic Hardware. *Front. Neuroanat.* 10 (2016), 37. DOI:http://dx.doi.org/10.3389/fnana.2016.00037

Christof Koch. 2004. *Biophysics of computation: information processing in single neurons*, Oxford university press.

Konrad Paul Kording. 2014. Bayesian statistics: Relevant for the brain? *Curr. Opin. Neurobiol.* 25 (2014), 130–133. DOI:http://dx.doi.org/10.1016/j.conb.2014.01.003

Duygu Kuzum, Rakesh G.D. Jeyasingh, Byoungil Lee, and H.S. Philip Wong. 2012. Nanoelectronic programmable synapses based on phase change materials for brain-inspired computing. *Nano Lett.* 12, 5 (2012), 2179–2186. DOI:http://dx.doi.org/10.1021/nl201040y

Quoc V. Le, Will Y. Zou, Serena Y. Yeung, and Andrew Y. Ng. 2011. Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 3361–3368. DOI:http://dx.doi.org/10.1109/CVPR.2011.5995496

Tai Sing Lee and David Mumford. 2003. Hierarchical Bayesian inference in the visual cortex. *J. Opt. Soc. Am. A. Opt. Image Sci. Vis.* 20, 7 (2003), 1434–48. DOI:http://dx.doi.org/10.1364/JOSAA.20.001434

Shih-Chii Liu and Tobi Delbruck. 2010. Neuromorphic sensory systems. *Curr. Opin. Neurobiol.* 20, 3 (June 2010), 288–95. DOI:http://dx.doi.org/10.1016/j.conb.2010.03.007

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Matteo Causo, Lille 1, 2017

Rodolfo R. Llinás. 2003. The contribution of Santiago Ramón y Cajal to functional neuroscience. *Nat. Rev. Neurosci.* 4, 1 (2003), 77–80. DOI:http://dx.doi.org/10.1038/nrn1011

Wolfgang Maass. 1997. Networks of spiking neurons: The third generation of neural network models. *Neural Networks* 10, 9 (1997), 1659–1671. DOI:http://dx.doi.org/10.1016/S0893-6080(97)00011-7

Stéphane Mallat. 2016. Understanding Deep Convolutional Networks. *arXiv* (2016), 1–17. DOI:http://dx.doi.org/10.1063/1.4939230

H. Markram, Joachim Lubke, M. Frotscher, and Bert Sakmann. 1997. Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs. *Science (80-.).* 275, 5297 (1997), 213–215. DOI:http://dx.doi.org/10.1126/science.275.5297.213

Andrew McAfee and Erik Brynjolfsson. 2012. Big Data. The management revolution. *Harvard Buiness Rev.* 90, 10 (2012), 61–68. DOI:http://dx.doi.org/10.1007/s12599-013-0249-5

Carver Mead and Mohammed Ismail. 1989. *Analog VLSI implementation of neural systems*, Springer Science & Business Media.

Bernhard Nessler, Michael Pfeiffer, Lars Buesing, and Wolfgang Maass. 2013. Bayesian Computation Emerges in Generic Cortical Microcircuits through Spike-Timing-Dependent Plasticity. *PLoS Comput. Biol.* 9, 4 (April 2013), e1003037. DOI:http://dx.doi.org/10.1371/journal.pcbi.1003037

Charles R. Noback, Norman L. Strominger, Robert J. Demarest, and David A. Ruggiero. 2005. *The Human Nervous System*, Totowa, NJ: Humana Press.

Dejan Pecevski and Wolfgang Maass. 2016. Learning probabilistic inference through STDP. *eneuro* (2016), ENEURO--0048.

Jan M. Rabaey. 2015. The human intranet - Where swarms and humans meet. *IEEE Pervasive Comput.* 14, 1 (2015), 78–83. DOI:http://dx.doi.org/10.1109/MPRV.2015.20

Rajat Raina, A. Madhavan, and AY Ng. 2009. Large-scale deep unsupervised learning using graphics processors. *ICML* (2009).

Sriram Ramgopal et al. 2014. Seizure detection, seizure prediction, and closed-loop warning systems in epilepsy. *Epilepsy Behav.* 37 (2014), 291–307. DOI:http://dx.doi.org/10.1016/j.yebeh.2014.06.023

Rajesh P.N. Rao and Dana H. Ballard. 1999. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nat. Neurosci.* 2, 1 (1999), 79–87. DOI:http://dx.doi.org/10.1038/4580

Jürgen Schmidhuber. 2015. Deep Learning in neural networks: An overview. *Neural Networks* 61 (2015), 85–117. DOI:http://dx.doi.org/10.1016/j.neunet.2014.09.003

Naresh R. Shanbhag et al. 2008. The search for alternative computational paradigms. *IEEE Des. Test Comput.* 25, 4 (2008), 334–343. DOI:http://dx.doi.org/10.1109/MDT.2008.113

Dmitri B. Strukov, Gregory S. Snider, Duncan R. Stewart, and R. Stanley Williams. 2008. The missing memristor found. *Nature* 453, 7191 (2008), 80–3. DOI:http://dx.doi.org/10.1038/nature06932

Massimiliano Di Ventra and Yuriy V. Pershin. 2013. The parallel approach. *Nat. Phys.* 9, 4 (2013), 200–202. DOI:http://dx.doi.org/10.1038/nphys2566

Massimiliano Di Ventra, Yuriy V. Pershin, and Leon O. Chua. 2009. Circuit elements with memory: Memristors, memcapacitors, and meminductors. *Proc. IEEE* 97, 10 (2009), 1717–1724. DOI:http://dx.doi.org/10.1109/JPROC.2009.2021077

M. Mitchell Waldrop. 2016. The chips are down for Moore's law. *Nat. News* 530, 7589 (2016), 144. DOI:http://dx.doi.org/10.1038/530144a

David A. Wells. 1889. *Recent Economic Changes, and Their Effect on the Production and Distribution of Wealth and the Well-being of Society*, D. Appleton.

Gerald Westheimer. 2008. Was Helmholtz a Bayesian? *Perception* 37, 5 (2008), 642–650. DOI:http://dx.doi.org/10.1068/p5973

Xi Zhou and Yaoyao Luo. 2013. Implementation of Hierarchical Temporal Memory on a Many-core Architecture. (2013).

Brian Zingg et al. 2014. Neural networks of the mouse neocortex. *Cell* 156, 5 (2014), 1096–1111. DOI:http://dx.doi.org/10.1016/j.cell.2014.02.023

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Matteo Causo, Lille 1, 2017

Colrain M. Zuppo. 2012. Defining ICT in a Boundaryless World: The Development of a Working Hierarchy. *Int. J. Manag. Inf. Technol.* 4, 3 (2012), 13–22. DOI:http://dx.doi.org/10.5121/ijmit.2012.4302

Abdullah M. Zyarah. 2015. Design and analysis of a reconfigurable hierarchical temporal memory architecture. (2015).

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Chapter 2 Background and State-of-the-Art

In this Chapter the reader is provided with a background on Bayesian Machine Learning (BML) and its state-of-the-art. Our choice for the BML algorithm is broadly motivated by illustrative examples and placed in the Machine Learning context. The probabilistic model generally employed for IoT and medical applications is introduced. Finally, the limitations of the state-of-the-art solutions are studied, leading to a more precise reformulation of our global approach.

2.1 Bayesian Machine Learning (BML)

Data-centric computing paradigms are all about data, as explained in Section 1.2. Full knowledge on the observed world involves knowledge on the phenomena generating the observations. However, these phenomena are generally hidden and both data and their observation can be noisy, partial or underlining chaotic and complex processes.

Probabilistic Machine Learning uses the rich language of Probability Theory to express coherent reasoning on all these uncertainties and learns a highly representative probabilistic model of the observed world for mining knowledge on hidden phenomena; conveniently processing current data and accurately forecasting future trends and developments [Ghahramani 2015].

The probabilistic model consists of random variables such as input data $\boldsymbol{X}$ and parameters $\boldsymbol{\theta}$; probability distributions, namely prior $p(\boldsymbol{\theta})$, posterior $p(\boldsymbol{\theta}|\boldsymbol{X})$, likelihood $p(\boldsymbol{X}|\boldsymbol{\theta})$ and marginal likelihood $p(\boldsymbol{X})$ and independence assumptions, i.e. causality relations between random variables .

In the Bayesian framework for inference, the Bayes' rule in Equation 2:1 transforms the prior into a posterior after having observed the input data. The *prior* distribution expresses a belief about the model parameters' initial value, while the resulting *posterior* distribution leads to properly accounting for uncertainty, predictions and decisions. Bayesian inference is applied to Machine Learning leading to the Bayesian Machine Learning (BML), when the model distributions are not fully specified and parameters are learnt on the basis of data [Barber 2011].

$$posterior = \frac{prior \cdot likelihood}{marginal\ likelihood}\ ; \qquad p(\boldsymbol{\theta}|\boldsymbol{X}) = \frac{p(\boldsymbol{\theta}) \cdot p(\boldsymbol{X}|\boldsymbol{\theta})}{p(\boldsymbol{X})}$$

Equation 2:1 – Bayes' rule

Suppose to toss a coin and register the binary result $X = \{H, T\}$, i.e. either a head H or a tail T, to test if the coin is fair. $\theta = 0.5$ represents the coin's bias and $p(\theta = 0.5) = 0.95$ our prior, i.e. at the beginning we strongly believe the coin is fair. The likelihood function underlies the Bernoulli distribution $p(X|\theta) = \theta^{N_H}(1-\theta)^{N_T}$, where N_H and $N_T \in [0,1]$ encode respectively the number of occurrence of heads or tails and $N_T = 1 - N_H$. The marginal likelihood is constant to $p(X) = 0.5$. Therefore, the essence of the Bayes' rule is to smooth or strengthen the initial prior and result into an accurate posterior after having been exposed to evidences.

In BML, $\boldsymbol{\theta}$ is generally unknown, i.e. hidden, and is intuitively estimated as the argument that maximizes the posterior. This is practically the Maximum A Posteriori (MAP) estimate in Equation 2:2.

$$\boldsymbol{\theta}^{MAP} = \underset{\boldsymbol{\theta}}{argmax}\ p(\boldsymbol{\theta}|\boldsymbol{X}) = \underset{\boldsymbol{\theta}}{argmax}\ p(\boldsymbol{\theta})p(\boldsymbol{X}|\boldsymbol{\theta})$$

Equation 2:2 – Maximum A Posteriori (MAP) estimate

Finally, if a constant prior is assumed, $\boldsymbol{\theta}$ is conveniently estimated as the argument that maximizes the likelihood and referred as Maximum Likelihood estimate, Equation 2:3.

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{argmax}\ p(\boldsymbol{X}|\boldsymbol{\theta})$$

Equation 2:3 – Maximum Likelihood estimate

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Matteo Causo, Lille 1, 2017

2.1.1 Probabilistic Graphical Models (PGMs) and inference

Several tools have been developed in Probability Theory to help modeling and solve modeling problems. For instance, a Probabilistic Graphical Model (PGM) is used to visually depict a probabilistic model and its mathematical formalism by means of structural assumptions [Jordan and Weiss 1996].

Generally, PGMs use graph-based factorized representations for encoding a set of independences held in the underlined distribution over a multi-dimensional space. For example, Bayesian networks are Direct Acyclic Graphs (DAGs) whose nodes represent random variables and are associated with a probability distribution. In particular, filled nodes are observed. Arrows represent conditional dependencies between nodes: variables are conditionally independent if their nodes are not connected or if the *d*-separation[10] criterion holds between any two sets of nodes given a third one. Local independences are equivalent to global ones.

Figure 2:1 shows the Bayesian network for the coin-tossing example after T tosses and its equivalent plate representation. Plate representation is a method of representing variables that repeat in a graphical model. Instead of drawing each repeated variable individually, a plate is used to group those variables into a subgraph, and the number of repetitions of the subgraph is drawn on the plate. The subgraph is duplicated that many times, the variables in the subgraph are indexed by the repetition number, and any links that cross a plate boundary are replicated once for each subgraph repetition [Buntine 1994].

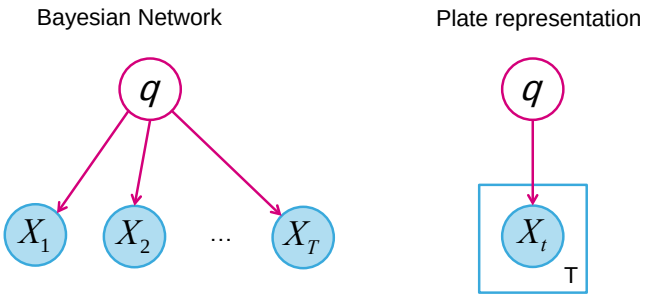


Figure 2:1 Bayesian Network for the coin tossing example and its plate representation.

The probabilistic model in Figure 2:1 is then expressed as the joint distribution of all random variables and results into the factorization of conditional distributions of Equation 2:4.

$$p(\theta, X_1, \dots, X_T) = p(\theta) \prod_{t=1}^{T} p(X_t|\theta)$$

Equation 2:4 – Chain rule for the Bayesian network in Figure 2:1.

PGMs are effective representation allowing to simplify the probabilistic approach and directly derive technics and algorithms for convenient data manipulation [Pearl 1988]. Inference, distribution and learning can be efficiently performed on PGMs through message passing algorithms, such as Belief Propagation or Revision [Jordan and Weiss 1996; Ghahramani 2002]. Furthermore, though Bayesian information fusion, PGMs can be combined on-the-fly. This is particularly useful in a typical IoT scenario, where knowledge extracted locally is shared among all connected devices whose own actions affect the whole system welfare.

The Bayes' rule in Equation 2:1 is trivially derived from the Bayesian Network in Figure 2:1 by reformulating the joint probability distribution in Equation 2:4 as $p(\theta, \boldsymbol{X}) = p(\boldsymbol{X})p(\theta|\boldsymbol{X})$, according to the conditional probability axiom [Okasha 2002]. By considering arrows connecting the model parameter θ and input data $\boldsymbol{X}$ as causality relations, an intuition on the Bayesian mechanisms for inference is gained. Indeed, the probabilistic model featured by the specific parameter θ represents the cause who generated the tossing results $\boldsymbol{X}$, i.e. the effects. Conversely, the inference reasoning the Bayes' rule underlies, proceeds bottom-up evaluating the cause according to the collected observations, i.e. the posterior $p(\theta|\boldsymbol{X})$.

[10] The d-separation criterion associates independence between two sets of variables with the existence of blocking paths between them, i.e. paths where arrows meet head-to-head at a node called collider. However, if the collider is a member of the conditioning set, or has a descendant in it, then it no longer blocks any path and variables become conditionally dependent.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

2.1.2 Expectation-Maximization (EM)

In the above formulation for the probabilistic model, the overall data have been assumed well specified. Generally, this is not the case because the information set $\boldsymbol{Y}$ may be incomplete [Roweis and Ghahramani 1999]. In fact, some input data may be missing and/or some model parameters $\boldsymbol{Z}$ are hidden, i.e. not observable. In this case, learning can be performed by a convenient and general-purpose iterative approach called Expectation-Maximization (EM), consisting of two phases [McLachlan and Krishnan 2008]. During Expectation, the algorithm completes the information set $\boldsymbol{Y} = \{\boldsymbol{X}, \boldsymbol{Z}\}$ by creating a lower bound for the likelihood, while during Maximization, the Maximum Likelihood estimate is computed for the lower bound.

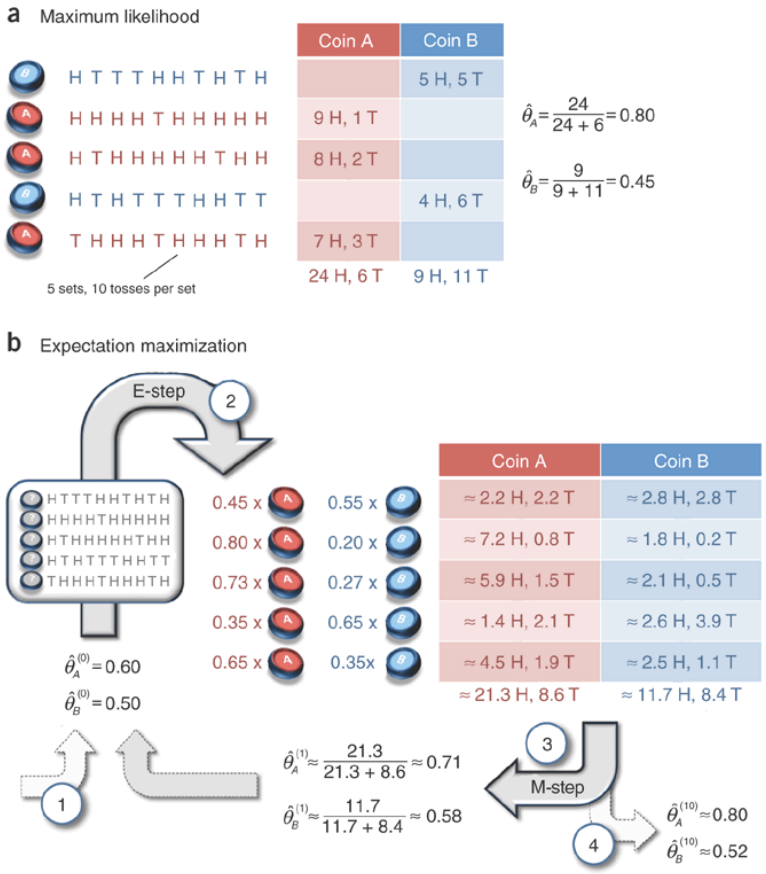


Figure 2:2 What is the expectation maximization algorithm? [Do and Batzoglou 2008]

((**a**) Maximum likelihood estimation. For each set of ten tosses, the maximum likelihood procedure accumulates the counts of heads and tails for coins A and B separately. These counts are then used to estimate the coin biases. (**b**) Expectation-Maximization. 1. EM starts with an initial guess of the parameters. 2. In the E-step, a probability distribution over possible completions is computed using the current parameters. The counts shown in the table are the expected numbers of heads and tails according to this distribution. 3. In the M-step, new parameters are determined using the current completions. 4. After several repetitions of the E-step and M-step, the algorithm converges.)

A simple example, proposed in [Do and Batzoglou 2008], follows from the coin-tossing experiment introduced in the previous sections and is illustrated in Figure 2:2. Two coins A and B are tossed in five sets S of ten tosses each, by arbitrarily choosing a coin per set. In this case $\boldsymbol{\theta} = \{\theta_A, \theta_B\}$, thus two biases have to be estimated. In Figure 2:2a, the information set is complete and so the Maximum Likelihood estimate in Equation 2:3 properly estimates both the model parameters $\boldsymbol{\theta}$ from input data. In Figure 2:2b, instead, the information set is incomplete because there is no information on $\boldsymbol{Z}$ that encodes the information on which coin generated each set, i.e. $Z_S = \{A, B\}$ and $p(\boldsymbol{Z}) = 0.5$. $\boldsymbol{Z}$ is hidden and has to be learned from data, too. The EM algorithm is successfully employed to this aim: the Expectation step (E-Step) evaluates $p(\boldsymbol{Z}|\boldsymbol{X})$ and the Maximization step (M-Step) learns $\boldsymbol{\theta}$ accordingly. The EM is proven to always increase the likelihood, i.e. the more the iterations, the better the learning accuracy [Barber 2011, sec.11.2.2]. Generally, it stops either if the likelihood cannot be maximized anymore or parameters do not improve any longer. However, it is guaranteed to converge to a local maximum of the likelihood, not to the global one [Rabiner 1989].

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

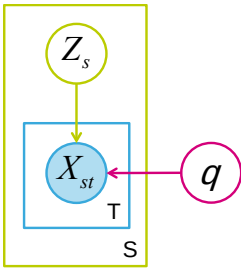


Figure 2:3 Bayesian network underling the probabilistic model for the example in Figure 2:2.

The EM is a really powerful tool however affected by two not negligible drawbacks, i.e. generalization and convergence. It handles the uncertainty on hidden information as long as it has full knowledge of its underlined probabilistic model. However, for many models of practical interest, the likelihood $p(\boldsymbol{X}, \boldsymbol{Z}|\boldsymbol{\theta})$ and the posterior distribution $p(\boldsymbol{\theta}, \boldsymbol{Z}|\boldsymbol{X})$ are usually hard to define or infeasible to evaluate because not analytically tractable in closed-form [Tzikas et al. 2008]. Furthermore, the Bayesian network associated with the coin-tossing example in Figure 2:2 and depicted in Figure 2:3 does not generalize to all possible applications. Therefore, we need a more powerful BML algorithm than EM, which is able to cope with uncertainty of input data and to automatically learn the full probabilistic model, i.e. parameters and structure of the PGM.

Regarding the EM convergence, in Section 5.2 we assume to have full knowledge of the probabilistic model of target applications and we gain insights on how to improve EM convergence and overall performance through distribution and Bayesian inference.

2.1.3 Variational Bayesian Expectation Maximization (VBEM)

The Variational Expectation-Maximization (VEM) algorithm has been proposed as a valid approximation scheme to bypass EM limitations [Barber 2011, sec.11.2.1]. In practice, the true posterior distribution is analytically approximated by a convenient factorized posterior $q(\boldsymbol{\theta}, \boldsymbol{Z})$, which is guaranteed to always define a lower bound for it [Jordan et al. 1999]. In the specific application to probabilistic inference, no other assumptions than factorization are made about the variational posterior distribution, indeed no restriction on the functional forms of individual factors is placed [Maindonald 2007, sec.10.1.1]. Convergence of VEM is guaranteed [Boyd and Vandenberghe 2004] and it results generally valid and applicable.

The Variational Bayesian Expectation-Maximization (VBEM) enriches the VEM with the Bayesian model selection, i.e. it is able to learn not just the model parameters, but also the model itself from the complete information set [Ghahramani and Beal 2001]. The Bayesian model selection consists of inferring a set of conditional independence statements between the model variables and choosing the cardinality of each discrete hidden variable or the dimensionality of the real-valued vectors of hidden variables [Beal 2003]. It relies on the Bayes' rule in Equation 2:5 and treats the model $\boldsymbol{m}$ as unknown parameter to be inferred during the Variational Bayesian Expectation (VBE) step. Therefore, the information set is completed and the posterior in Equation 2:1 can be computed as in Equation 2:6.

$$p(\boldsymbol{m}|\boldsymbol{Y}) = \frac{p(\boldsymbol{m}) \cdot p(\boldsymbol{Y}|\boldsymbol{m})}{p(\boldsymbol{Y})}$$

Equation 2:5 – Bayesian model selection.

$$p(\boldsymbol{\theta}|\boldsymbol{Y}, \boldsymbol{m}) = \frac{p(\boldsymbol{\theta}|\boldsymbol{m}) \cdot p(\boldsymbol{Y}|\boldsymbol{\theta}, \boldsymbol{m})}{p(\boldsymbol{Y}|\boldsymbol{m})}$$

Equation 2:6 – Posterior over the complete information set.

$$p(\boldsymbol{Y}|\boldsymbol{m}) = \sum_{\boldsymbol{\theta}} p(\boldsymbol{\theta}|\boldsymbol{m}) \cdot p(\boldsymbol{Y}|\boldsymbol{\theta}, \boldsymbol{m}) \quad ; \qquad p(\boldsymbol{Y}|\boldsymbol{m}) = \int d\boldsymbol{\theta} \cdot p(\boldsymbol{\theta}|\boldsymbol{m}) \cdot p(\boldsymbol{Y}|\boldsymbol{\theta}, \boldsymbol{m})$$

Equation 2:7 – Marginal likelihood for the complete information set.

The marginal likelihood $p(\boldsymbol{Y}|\boldsymbol{m})$ in Equation 2:7 is crucial for Bayesian model selection because incorporates a model complexity penalty by summing/integrating over all model parameters. Therefore, it embodies the Occam's razor effect to prevent over-fitting [Rasmussen and Ghahramani 2001; Ghahramani and Beal 2001; Barber 2011, sec.12.3].

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Matteo Causo, Lille 1, 2017

To understand the Bayesian model selection mechanism and to intuitively compare the three BML approaches to learning introduced in this Chapter, namely Maximum Likelihood estimate, EM and VBEM, we compare learning to a treasure hunt. The sought treasure is a concept or knowledge we finally get if we successfully gather all the clues hidden in several tests we encounter by following the path indicated in a map. If the map is given and so are the solutions to all tests, the treasure hunt is only a matter of getting to the treasure. This is a rare case if not impossible in reality and represents the Maximum Likelihood estimate. In a more realistic case, only the map is given, but no clues. Therefore, we firstly have to pass all tests in the map to collect all the clues, and then get to the treasure and try to open the treasure chest with them. If it does not open, then some of the tests have not been successful. We have to try again and again until all the clues are correctly known. This case represents the EM. Finally, in the reality of cases neither the map nor the clues are given, but we want to get the treasure as well. The Bayesian model selection mechanism the VBEM incorporates, firstly looks for truthful maps among infinite scams and then triggers the treasure hunt for an essential subset of them.

The VBEM is astonishingly powerful, but it aims to a very ambitious task. This raises the questions on whether it is practically feasible and what is the cost to pay for its exhaustive search. As introduced in Section 2.1, the Bayesian treatment proposes a posterior by adapting a prior on the basis of evidences. Therefore, without a prior, i.e. an initial belief on the sought knowledge, the model cannot be well defined and the algorithm cannot converge. This is true both for the VBEM and the EM. In fact, the latter needs priors on the right solutions to all tests and the former needs priors both for tests and maps. Even very vague priors beliefs can be useful. Indeed, given enough data, the effect of the prior is generally overcome by the likelihood and posterior conclusions will converge around reasonable models [Ghahramani 2013]. Therefore, the key ingredient of Bayesian methods is not the prior itself but the idea of averaging over different possibilities. For this reason, the singularities that arise in Maximum Likelihood are absent in the Bayesian treatment [Maindonald 2007]. Finally, as the VBEM generalizes the EM under the assumption of a constant prior [Barber 2011, sec.11.5.1], the cost of its computational overhead is however leveled by its substantial advantages.

Chapter 3 is devoted to simplify and enhance the VBEM algorithm in order to reduce its computational overhead and make its hardware implementation both feasible and attractive so that to fulfill this PhD thesis objective as described in Section 1.4.

2.1.4 Hidden Markov Model Gaussian Mixture Model (HMM GMM)

Our objective is to approach Neuro-Inspired computing with an Ultra-Low-Power (ULP) prospective, so that to provide IoT and medical applications with intelligent processing as close as possible to data sources. Therefore, targeted applications involve real-time stream analytics such as audio, video and biosignals processing.

In the analysis of continuous streams, the underlined model implies relations between consequent data, i.e. they are not independent and identically distributed (i.i.d.), but sequential data. Learning the relations between past $(X_1, \dots, X_{t-1})$ and present (X_t) observations is of paramount interest because they can be exploited for predicting future ones. A probabilistic representation of such model is given in Figure 2:4 and the joint probability distribution in Equation 2:8 results from the chain rule in Equation 2:4.

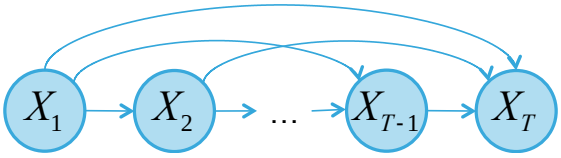


Figure 2:4 Bayesian network for sequential data.

$$p(X_1, \dots, X_T) = \prod_{t=1}^{T} p(X_t \mid X_1, \dots, X_{t-1})$$

Equation 2:8 – Joint distribution for the sequential data model in Figure 2:4.

However, it would be impractical to consider a general dependence of future observations on all the previous ones because the complexity of such a model would grow unlimited as the number of observations increases. Intuitively, recent observations are likely to be more informative than older ones in predicting future values. Indeed, the Markov property assumes that future predictions are independent of all but the most recent ones. In particular, the first-order Markov chain relates the present observation (X_t) only to the previous one (X_{t-1}). In this way, however, the richness of all those causal relationships among observations is lost. In a speech recognition application, if each observation is an elementary phoneme, associating it to the previous one is interesting, but it would be better to bind it to the three or four preceding phonemes in order to seize entire words. By introducing a hidden variable Z_t for each observation X_t, the resulting Bayesian network in Figure 2:5 is a Hidden Markov Model (HMM). It is not limited by the Markov

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

assumption and relies on a limited number of parameters. In particular, the first-order Markov chain on hidden variables $\boldsymbol{Z}$ satisfies the Markov assumption. In fact, it meets the key conditional independence property $Z_{t+1} \coprod Z_{t-1} | Z_t$ and given Z_t, X_t results i.i.d. with respect to $\boldsymbol{Z}$ and $\boldsymbol{X}$ at all other time indices because of the *d*-separation criterion [Ghahramani 2001]. However, if no $\boldsymbol{Z}$ is observed, there is always an unblocked path connecting any observations $\boldsymbol{X}$ [Maindonald 2007, sec.13.1]. Thus the current observation X_t does not exhibit any conditional independence properties but depends on all the previous observations $\boldsymbol{X}$ as they do not satisfy the Markov property at any order.

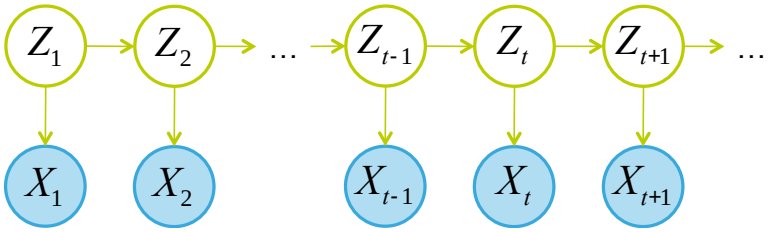


Figure 2:5 Bayesian network for a Hidden Markov Model (HMM)

(PGM representing the state space model, i.e. the structure of an HMM. The input data $\boldsymbol{X}$ are observed so they are filled. The hidden variables $\boldsymbol{Z}$ are not observed and allow for the independence between past states and observations.)

Generally, each hidden variable Z_t is chosen to be a N-dimensional binary indicator with a 1-of-N coding scheme, i.e. it is always at zero, but for the state generating the corresponding observation X_t. The probability distribution of Z_t depends on the state of the previous hidden variable Z_{t-1} through the conditional distribution $p(Z_t | Z_{t-1})$, i.e. the transition probability summarized in the transition matrix $\boldsymbol{A}$. The initial hidden variable Z_1 has not parent nodes and so its probability distribution $p(Z_1)$ is represented by the initial value vector $\boldsymbol{\pi}$. Finally, the conditional distribution of the observed variable $p(X_t | Z_t, \boldsymbol{\theta})$ is known as emission probability, it is summarized in the emission matrix $\boldsymbol{B}$ and it is governed by a set of parameters $\boldsymbol{\theta}$. Therefore, a HMM is completely defined by the joint distribution in Equation 2:9 and the parameters $\boldsymbol{\phi} = \{\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{\theta}\}$ detailed in Table 2:1.

$$p(\boldsymbol{Y}|\boldsymbol{\phi}) = \boldsymbol{\pi}\left[\prod_{t=2}^{T} \boldsymbol{a}_t\right]\prod_{t=1}^{T} \boldsymbol{b}_t$$

Equation 2:9 – Joint distribution for the HMM in Figure 2:5.

Parameter	Definition	Description	Dimension
$\boldsymbol{\pi}$	$\pi_i \equiv p(Z_{1_i} = 1)$	Initial distribution	N
$\boldsymbol{A}$	$a_{t_{ji}} \equiv p(Z_{t_i} = 1 \| Z_{t-1_j} = 1)$	Transition distribution	$TxNxN$
$\boldsymbol{B}$	$b_{t_i} \equiv p(X_t \| Z_{t_i} = 1, \boldsymbol{\theta})$	Emission distribution	TxN
$\boldsymbol{\theta}$		Emission parameters	

Table 2:1 HMM parameters summary.

HMM can be applied to the coin-tossing example in Section 2.1.2 to take trace of the tossing history; accurately predict the next chosen coin and improve coins' biases estimation. Therefore, $\boldsymbol{\theta} = \{\theta_A, \theta_B\}$, $N = 2$ and the emission distribution results $b_t \equiv p(Z_{t_A} = 1)\left[\theta_A^{N_H}(1-\theta_A)^{N_T}\right] + p(Z_{t_B} = 1)\left[\theta_B^{N_H}(1-\theta_B)^{N_T}\right]$. Figure 2:6 shows the HMM transition diagram. As the experiment is carried out, the system evolves between the two possible states $\boldsymbol{Z}$ related to the current chosen coin. Transition probabilities are collected into the $\boldsymbol{A}$ matrix and the system initial condition into $\boldsymbol{\pi}$. Provided full knowledge of this model, it is possible to implement HTM continuous inferring and prediction processes along with its sequential memory and hierarchy principles.

Generally, the emission probability distribution is chosen to be a linear combination of K Gaussian components $\mathcal{N}_k(\boldsymbol{X}_t | \boldsymbol{\mu}_k, \boldsymbol{R}_k)$ that is a *universal approximator* for continuous densities [Carreira-Perpiñán 2000]. Furthermore, it can be formulated as the probabilistic model described by the likelihood distribution in Equation 2:10, known as Gaussian Mixture Model (GMM) and governed by the parameters $\boldsymbol{\theta} = \{\boldsymbol{\alpha}, \boldsymbol{\mu}, \boldsymbol{R}\}$, where $\boldsymbol{\alpha}$ are the mixing coefficients, $\boldsymbol{\mu}$ the Gaussian means and $\boldsymbol{R}$ the covariances.

$$p(\boldsymbol{X}|\boldsymbol{\theta}) = \prod_{t=1}^{T}\left[\sum_{\boldsymbol{L}_t} p(\boldsymbol{X}_t|\boldsymbol{L}_t, \boldsymbol{\theta})\right] = \prod_{t=1}^{T}\left[\sum_{\boldsymbol{L}_t} p(\boldsymbol{L}_t|\boldsymbol{\alpha}) p(\boldsymbol{X}_t|\boldsymbol{L}_t, \boldsymbol{\mu}, \boldsymbol{R})\right] = \prod_{t=1}^{T}\left[\sum_{k}^{K} \boldsymbol{\alpha}_k \mathcal{N}_k(\boldsymbol{X}_t|\boldsymbol{\mu}_k, \boldsymbol{R}_k)\right]$$

Equation 2:10 – Likelihood distribution for the Gaussian Mixture Model (GMM).

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

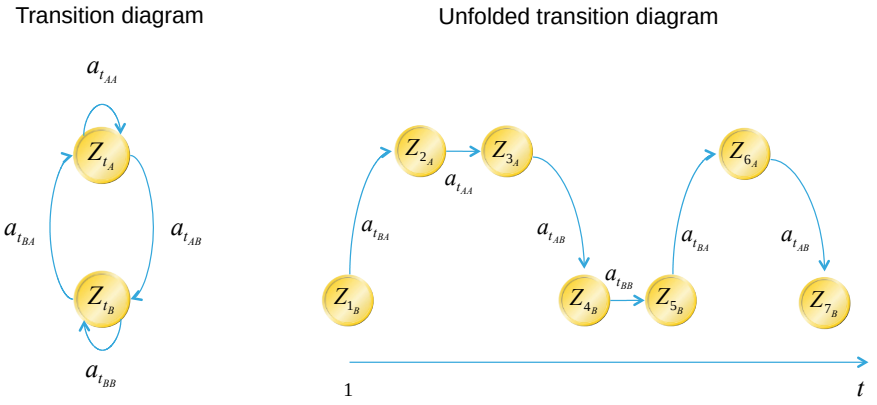


Figure 2:6 Transition diagram for the HMM associated with the coin-tossing example.

(Transition diagram (Left) and its unfolded form (Right) where each column corresponds to one of the latent variables Z_t.)

In fact, by introducing a K-dimensional binary hidden indicator $\boldsymbol{L}_t$ with a 1-of-K representation we obtain $p(\boldsymbol{L}_t|\boldsymbol{\alpha}) = \prod_k^K \alpha_k^{\boldsymbol{L}_{t,k}}$ and $p(\boldsymbol{X}_t|\boldsymbol{L}_t, \boldsymbol{\mu}, \boldsymbol{R}) = \prod_k^K \mathcal{N}_k(\boldsymbol{X}_t|\boldsymbol{\mu}_k, \boldsymbol{R}_k)^{\boldsymbol{L}_{t,k}}$.

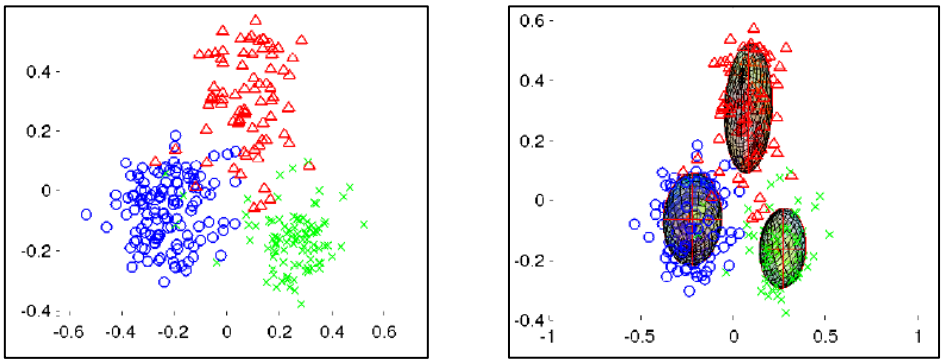


Figure 2:7 The speaker diarization problem addressed in [Shum et al. 2013].

(Input data collected for a three-speaker conversation are segmented (Left) and clustered (Right). Left) The first two principal components of input data are segmented to distinguish among speakers according to their color. Right) GMM 3D clusters collect together data relative to the same speaker.)

Practically, GMM is particularly adapted for image segmentation, audio and biosignals streaming analysis applications [Reynolds 2008; Alam et al. 2014; Motlicek et al. 2015; Nguyen and Wu 2013; Mekhalfa et al. 2007; Balafar 2012]. For instance in [Shum et al. 2013] the speaker diarization problem is addressed and input data are conveniently explained with a GMM, as shown in Figure 2:7. Speaker diarization, a.k.a. the *who spoke when* problem, consists in *segmenting*, i.e. annotating an unlabeled audio file involving several speakers and *clustering*, i.e. associating the different segments of speech belonging to the same speaker [Tranter and Reynolds 2006], as shown in Figure 2:7Left where the first two principal components of a three-speaker conversation are reported after having applied Principal Component Analysis (PCA). The GMM perfectly explains those data by means of 3D Gaussian components in Figure 2:7Right, with specific means and covariances for each speaker. Otherwise, if input data are not well explained by the Gaussian framework, preprocessing is employed to obtain Gaussian-friendly observations.

Thanks to modularity of PGMs, the two models can be easily combined into an HMM GMM as in Figure 2:8. Therefore, an HMM GMM is exhaustively described by the joint probability distribution over the complete set $\boldsymbol{Y} = \{\boldsymbol{X}, \boldsymbol{Z}, \boldsymbol{L}\}$ in Equation 2:9 and the parameters $\boldsymbol{\phi} = \{\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{\alpha}, \boldsymbol{\mu}, \boldsymbol{R}\}$ detailed in Table 2:2, where D is the input data dimensionality.

Parameter	Definition	Description	Dimension
$\boldsymbol{\pi}$	$\pi_i \equiv p(Z_{1_i} = 1)$	Initial distribution	N
$\boldsymbol{A}$	$a_{t_{ji}} \equiv p(Z_{t_i} = 1 \mid Z_{t-1_j} = 1)$	Transition distribution	$TxNxN$
$\boldsymbol{B}$	$b_{t_i} \equiv p(L_{t_i} = 1 \mid Z_{t_i} = 1) p(X_t \mid L_{t_i} = 1, \boldsymbol{\theta})$	Emission distribution	$TxNxK$
$\boldsymbol{\alpha}$		Gaussian mixing coefficients	NxK
$\boldsymbol{\mu}$		Gaussian means	$NxKxD$

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

$\boldsymbol{R}$	Gaussian covariances	$NxKxDxD$

Table 2:2 HMM GMM parameters summary.

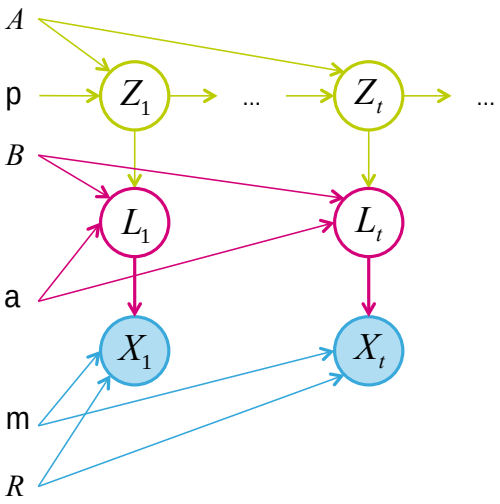


Figure 2:8 Bayesian network for the HMM GMM.

This model is particularly interesting for studying and modeling continuous streams. In fact, it incorporates the HMM characteristics of continuous prediction and sequential memory and creates a hierarchical structure through the Bayesian network in Figure 2:8 on which inference can be efficiently implemented. So many of the HTM principles described in Section 1.3.4, are naturally incorporated in the HMM GMM, which is the model of reference for the present PhD thesis work. Finally, learning is also implemented with the application of the VBEM to the HMM GMM.

2.2 Position of the present PhD thesis

The VBEM applied to the HMM GMM is a well-known state-of-the-art powerful tool, while solutions that make it nonparametric and capable of on-line and distributed elaboration are matter of ongoing research. Nevertheless, no focus has ever been devoted on ULP-oriented integration with today's technology in order to maximally exploit the benefits of an HW and algorithmic co-optimization that is essential for popularizing Machine Learning solutions over portable and wearable electronics [HALO 2015]. Therefore, by adopting and optimizing the VBEM HMM GMM we can replicate HTM principles and theory with an ULP prospective and achieve this PhD thesis objective according to the main idea in Section 1.5.

2.2.1 Review of ML algorithms and methods

In this Section, we aim to place the VBEM in the ML context and to reveal its connections with the most popular Machine Learning algorithms.

In the decade around 2000, many practical and commercial pattern recognition applications were dominated by kernel methods like the Support Vector Machine (SVM) [Schmidhuber 2015]. Compared with linear models for regression and classification that are limited by the curse of dimensionality, the SVM scales well for large problems. In fact, it defines basis functions that are centered on the training data points and then selects a subset of them during training, instead of adapting them to the data. Thus, the resulting model is generally much smaller than the number of training points, although it typically increases with the size of the training set. Furthermore, the training involves nonlinear optimization, but the determination of the model parameters corresponds to a convex optimization problem, and so any local solution is also a global optimum.

An alternative approach is to use parametric forms for the basis functions in which parameters are adapted during training. The most successful model in the context of pattern recognition is the Artificial Neural Network (ANN), where the network parameters, i.e. weights and biases, are determined with the Maximum Likelihood estimate as in Equation 2:3 efficiently solved by error Back-Propagation (BP). For many applications, the resulting model can be significantly more compact, and hence faster to evaluate than with a SVM having the same generalization performance. The price to be paid is that the likelihood function is no longer convex.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Matteo Causo, Lille 1, 2017

Approaches presented so far are referred as parametric methods because they use a fixed number of parameters and so they might result in a poor model of the distribution generating the data. Nonparametric models, instead, constitute an approach to model selection and adaptation, where models are allowed to grow with data. Nonparametric frequentist approaches, such as K-nearest-neighbor, kernel density estimator and histograms, require the entire training dataset to be stored, leading to large memories and expensive computations to access it. A two-layer ANN is a reasonable nonparametric model [Neal 1996] that can approximate any given function with arbitrary accuracy if the number of hidden units is sufficiently large. However, in the Maximum Likelihood framework, the number of hidden units needs to be limited according to the size of the training set in order to avoid over-fitting. A Bayesian treatment overcomes this limitation.

Bayesian nonparametric models use only a finite subset of the available parameters to explain a finite sample of observations, such that the effective complexity of the model adapts to the data [Orbanz and Teh 2005]. Classical adaptive problems, such as nonparametric estimation and model selection, can thus be formulated as Bayesian inference problems, as discussed above. In the case of a multilayered ANN, the highly nonlinear dependence of the network function on the parameter values do not allow for an exact Bayesian treatment. So, approximation of the posterior distribution has to be applied.

There are two classes of approximation schemes. Stochastic approximation techniques based on numerical sampling, such as Markov Chain Monte Carlo (MCMC), can generate exact results given infinite computational resources. In practice, sampling methods can be computationally demanding, often limiting their use to small problems. Conversely, deterministic approximation schemes like Variational Bayesian inference, as in the case of VBEM, scale well to large applications. They are based on analytical approximations to the posterior distribution, so they can never generate exact results.

ANNs with Variational Bayesian inference approximation offer a probabilistic interpretation of Deep Learning (DL) models by inferring distributions over the models' weights [Gal and Ghahramani 2015a]. DL has allowed ANNs to succeed in many image recognition and classification competitions where predictions should be invariant under one or more transformations of the input variables. ANNs can learn the invariance if sufficiently large numbers of training patterns are available. In fact, with the advent of more powerful machines, ANNs have prevailed over SVMs. However, thanks to DL the invariance property has been built into the structure of the ANNs allowing for even better performance. Invariance could also be built into the preprocessing by extracting features that are invariant under the required transformations, so that there is no more the requirement for a huge dataset for training.

The VBEM is a Bayesian approach and provides an elegant way of trading off the complexity of the model and the complexity of the data. It also gives reasonable answers not just for big data sets, but also for small ones [Ghahramani 2015]. Finally, it scales well to large applications and does not need huge training if an external preprocessing is provided, meaning that an ultra-low-power solution could be obtained and can be placed as close as possible to data sources.

2.2.2 State-of-the-Art on VBEM HMM GMM

The VBEM has been successfully applied to the HMM GMM [MacKay 1997; Ji et al. 2006]. The VBEM HMM GMM learns the model parameters and structure by automatically determining the number of states N and the number of Gaussian components K. In practice, it is capable of learning the exact probabilistic model on the basis of the input data, i.e. to dynamically size $\boldsymbol{Z}$, $\boldsymbol{L}$ and all other model parameters in the probabilistic world.

In the VBEM HMM GMM formulation, the posterior distribution $p(\boldsymbol{Z}, \boldsymbol{L}, \boldsymbol{\phi}|\boldsymbol{X})$ for the HMM GMM in Figure 2:8 is approximated with the variational distribution $q(\boldsymbol{Z}, \boldsymbol{L}, \boldsymbol{\phi})$ in Equation 2:11.

$$p(\boldsymbol{Z}, \boldsymbol{L}, \boldsymbol{\phi}|\boldsymbol{X}) \approx q(\boldsymbol{Z}, \boldsymbol{L}, \boldsymbol{\phi}) = q(\boldsymbol{Z}, \boldsymbol{L})q(\boldsymbol{\phi})$$

Equation 2:11 – Factorized Variational distribution for the VBEM HMM GMM [Ji et al. 2006].

A measure of the difference between the true posterior p and its tractable approximation q is the Kullback-Leibler (KL) divergence that is by construction nonnegative and zero only for identical distributions, i.e. $KL(p||q) \geq 0$. In particular, the fundamental relation of the mean field approximation, i.e. $\log p(\boldsymbol{X}) = F(q) + KL(p||q)$ indicates that the negative free energy term $F(q)$ is a lower bound on the log-marginal likelihood, i.e. $\log p(\boldsymbol{X}) \geq F(q)$. Therefore, the aim of the VBEM is to maximize this lower bound by tuning the variational posterior $q(\boldsymbol{Z}, \boldsymbol{L}, \boldsymbol{\phi})$ such that it approaches the true posterior and minimizes the KL divergence. This goal is achieved by conveniently iterating between two alternative steps: the Variaitonal Bayesian Expectation (VBE) and the Variational Bayesian Maximization (VBM). Figure 2:9 helps visualizing the VBEM procedure applied to input data with a similar distribution to Figure 2:7.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Variational Bayesian Expectation (VBE). With the variational posterior on model parameters fixed at $q(\boldsymbol{\phi})$, update the variational posterior on hidden variables $q(\boldsymbol{Z}, \boldsymbol{L})$ to maximize $F(q)$ and get the tightest lower bound on $\log p(\boldsymbol{X})$. This corresponds in completing the information set $\boldsymbol{Y}$ and choosing the probabilistic model that better explains the input data among a set of possible models, Figure 2:9Left.

Variational Bayesian Maximization (VBM). With the variational posterior on hidden variables fixed at $q(\boldsymbol{Z}, \boldsymbol{L})$, update the variational posterior on model parameters $q(\boldsymbol{\phi})$ to maximize both $F(q)$ and $\log p(\boldsymbol{X})$. The chosen model is successively fine tuned to better adapt to the input data, Figure 2:9Right.

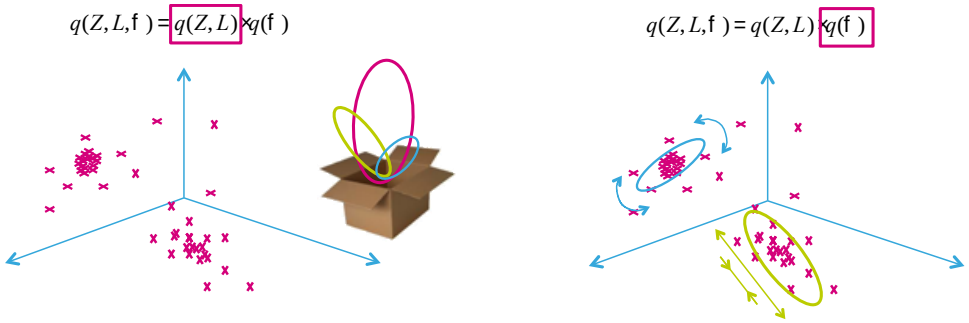


Figure 2:9 An intuitive illustration of the VBEM procedure.

(Input data in a 3D-space representation (Left) are assigned with the model that better explains them during the Variational Bayesian Expectation (VBE) step. This model (Right) is successively adjusted to better adapt to the input data during the Variational Bayesian Maximization (VBM) step. In both cases the VBEM consists into a maximization procedure targeting the variational distribution in the fuchsia rectangle.)

As Bayesian approach, the VBEM requires prior distributions over the model parameters $\boldsymbol{\phi}$. A natural choice for the prior over $\boldsymbol{\pi}$, the rows of $\boldsymbol{A}$ and the rows of $\boldsymbol{B}$ is the Dirichlet distribution because it is the conjugate prior over the multinomial distribution. Similarly, the Normal-Wishart (NW) distribution is the prior over the Gaussian distribution. Thus, the prior on the model parameters can be expressed as $p(\boldsymbol{\phi}) = p(\boldsymbol{\pi})p(\boldsymbol{A})p(\boldsymbol{B})p(\boldsymbol{\theta})$ and is detailed in Equation 2:12.

$$p(\boldsymbol{\pi}) = Dir(\pi_1, \dots, \pi_N | u_1^{\pi}, \dots, u_N^{\pi})$$

$$p(\boldsymbol{A}) = \prod_{n=1}^{N} Dir(A_{n1}, \dots, A_{nN} | u_{n1}^{A}, \dots, u_{nN}^{A})$$

$$p(\boldsymbol{B}) = \prod_{n=1}^{N} Dir(B_{n1}, \dots, B_{nK} | u_{n1}^{B}, \dots, u_{nK}^{B})$$

$$p(\boldsymbol{\theta}) = \prod_{n=1}^{N} \prod_{k=1}^{K} NW(\mu_{ik}, R_{ik} | \alpha_{ik}, \beta_{ik}, \lambda_{ik}, m_{ik})$$

Equation 2:12 – Detailed priors on model parameters [Ji et al. 2006].

A block diagram of the VBEM HMM GMM is depicted in Figure 2:10. Input data could have any dimensionality D and after having been processed by the Feature Extraction Engine (FEE), they are presented to the Bayesian classifier. The FEE acts like HTM *Sensory Encoders* described in Section 4 and extracts important features that are invariant to input data transformations.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

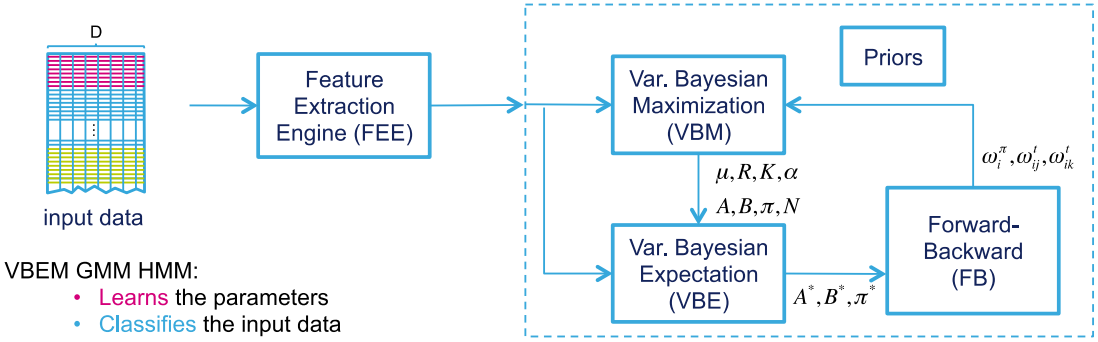


Figure 2:10 Block diagram of the state-of-the-art VBEM HMM GMM.

Maximization is efficiently computed trough inference on the HMM GMM PGM by using two-stage message passing algorithms, namely the Forward-Backward (FB) algorithm [Rabiner 1989; MacKay 1997] and the Forward Filtering Backward Smoothing (FFBS) algorithm [Ott 1967; Khreich et al. 2010]. Both lead to the exact marginal because the HMM GMM has a tree structure [Jordan 2003]. The sufficient statistics ω_i^{π}, ω_{ij}^{t}, and ω_{ik}^{t}, i.e. the smoothed conditional state probability densities, update the model parameters $\boldsymbol{\phi}^*$ and the approximate distribution $q_t^*(\boldsymbol{\phi}^*)$ of the optimized model.

The VBEM HMM GMM has been demonstrated to be a valid, flexible and powerful approach in several practical applications such as speaker diarization, epileptic seizure prediction and particle tracking [Valente and Wellekens 2005; Watanabe et al. 2006; Esmaeili et al. 2014; Martin et al. 2016]. The learning mechanisms the VBEM implements interestingly replicate those in HTM theory. In fact, through the continuous alternating between the two VBEM steps a closed-loop is implemented. The model is learnt from data and then it is applied to explain the data and to get the feedback, which improves on the model learning itself.

ALGORITHM 1. state-of-the-art VBEM HMM GMM [Barber 2011, sec.11.5]

Input: all input data $\boldsymbol{X}$.
Output: Classification result ω_i^h and HMM GMM parameters $\boldsymbol{\phi}^*$.
Priors: α_0; β_0; $\boldsymbol{m}_0$; $\boldsymbol{\lambda}_0$; $\boldsymbol{u}^{\pi}$; $\boldsymbol{u}^{A}$; $\boldsymbol{u}^{B}$

while $\boldsymbol{\phi}$ change **do**
$t \leftarrow t + 1$
$q_t^*(\boldsymbol{Z}, \boldsymbol{L}) = \arg\max_{q(\mathbf{Z},\mathbf{L})} F(q)$ // VBE HMM GMM
$q_t^*(\boldsymbol{\phi}^*) = \arg\max_{q(\boldsymbol{\phi})} F(q)$ // VBM HMM GMM
end

The state-of-the-art VBEM HMM GMM, in Algorithm 1, is software-based and to our knowledge there is no hardware implementation of it, yet. In fact, a naïve mapping of VBEM in a physical device for acceleration purpose is neither low-power nor scalable because of the required memory, computing power and overall complexity. The elaboration is performed in batch mode, i.e. data are first collected in their entirety and then processed. Furthermore, the whole computation is confined to a single powerful machine in a centralized manner. Therefore, the processing is ex-situ, i.e. far away from the data source. This solution results to be impractical for portable and wearable applications because of their limited power and memory resources: data are collected and transferred to the powerful machine or the Cloud with a consequent communication bottleneck.

Finally, the VBEM HMM GMM is an interesting and complete learning algorithm, by construction less demanding than DL, SVM and MCMC approaches in terms of required resources. The state-of-the-art VBEM HMM GMM embodies many of the HTM principles and theories, but it is batch, i.e. elaboration is massive on all the data and not on-line. In the next Subsections, we show, however, that it can benefit of an on-line and distributed formulation in order to overcome the communication bottlenecks and even reduce its complexity, thus a hardware acceleration will be feasible and attractive for a Learning-on-a-Chip implementation.

2.2.3 Bayesian Nonparametric

Probabilistic Machine Learning aims to learn the probabilistic model underlined in the observed reality. Learning the right model is crucial for explaining input data and correctly predicting future trends. However, Bayesian formulation treats the overall model, i.e. structure and parameters, as hidden variables to infer from the input data like every other unknown information. Although this makes

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Matteo Causo, Lille 1, 2017

the Bayesian framework a sought data-centric paradigm, its effectiveness can dumfound. How does it discern among different information looking to the same input data? How accurately can it model reality?

The employed mechanism is the Bayesian model selection. It is not a magic tool that solves every problem as it would seem, rather it is a working practical tool implementing a simple idea: the model is not actually learnt from data, but it is selected from a sequence of models of increasing complexity as the one with the right complexity to sufficiently explain the input data. Therefore, there is no over-fitting due to the Occam's razor effect and what is actually learnt is the required complexity in terms of number of hidden variables. For instance, the VBEM HMM GMM in Section 2.2.2 is a Bayesian Parametric model because it selects the number of HMM states and the number of Gaussian components among the possible N and K, respectively.

The modeling accuracy improves intuitively if the model complexity is not restricted, but flexibly extended to capture any possible dataset. In fact, Bayesian Nonparametric models start with potentially infinite complexity and reduce it until it properly models the real world. Indeed, they are neither under-fitting nor over-fitting input data.

The Chinese restaurant process explains Bayesian Nonparametric working principles and ability to cope with infinite complexity with a meaningful analogy [Aldous 1983]. Imagine an infinitely large restaurant containing an infinite number of tables, each of which served with a specific dish. This restaurant operates an unusual seating policy whereby new diners are seated either at a currently occupied table with probability proportional to the number of guests already seated there, or at an empty table with constant probability. Therefore, a delicious dish is likely to push more guests on the same table in a *rich get richer* fashion. Each table represents a cluster and its associated dish the common hidden cause attracting guests. The Chinese restaurant process is related to the Pólya urn sampling scheme [Blackwell and MacQueen 1973] as both implement a Dirichlet process. It is worth to mention a third method, the Stick-Breaking construction, actually used to prove the Dirichlet process existence [Sethuraman 1994].

In [Teh et al. 2006], Dirichlet processes have been organized into a hierarchy to identify common hidden causes among hidden variables and to allow for sharing common dependencies among clusters. This concept, spread to all Bayesian Nonparametric modeling, led to Hierarchical Bayesian Nonparametric models [Teh and Jordan 2010] such as infinite Hidden Markov Models (iHMM) [Beal et al. 2002], i.e. HMM where the number of HMM state N tends to infinity.

2.2.4 On-line elaboration

The state-of-the-art VBEM HMM GMM introduced in Section 2.2.2 performs in batch mode, i.e. data are firstly collected in their entirety and then processed as shown in Figure 2:10. Conversely in the on-line elaboration, input data are presented to the system in small consecutive sequences and progressively processed.

The on-line VBEM is able to dynamically adapt the model structure by exploiting the Bayesian model selection mechanism. Furthermore, it shows faster and better performance than the batch VBEM, where the estimation quality of the posterior probability for the hidden variables improves rather slowly especially for large amounts of data [Sato 2001].

A general on-line formulation for the VBEM has been provided in [Sato 2001] and it has been applied to the GMM [Cournapeau et al. 2010] and to the HMM with Gaussian observations [Koshinaka et al. 2009], but not to the overall HMM GMM.

Recently, in [Broderick et al. 2013] a Streaming Bayesian approach has been proposed. The input stream is partitioned into mini-batches, i.e. sequences $\boldsymbol{S}_h$ of T i.i.d. data, so that $\boldsymbol{S}_h = \{\boldsymbol{X}_1, \dots, \boldsymbol{X}_T\}$ and $p(\boldsymbol{S}_h|\boldsymbol{\phi}) = p(\boldsymbol{X}_1, \dots, \boldsymbol{X}_T|\boldsymbol{\phi}) = \prod_{t=1}^{T} p(\boldsymbol{X}_t|\boldsymbol{\phi})$. After having processed $h-1$ sequences, the posterior distribution for $\boldsymbol{S}_h$ is computed without revisiting old data according to the naïve Bayes' rule in Equation 2:13, whose prior is the posterior we previously computed for $\boldsymbol{S}_{h-1}$.

$$p(\boldsymbol{\phi}|\boldsymbol{S}_1, \dots, \boldsymbol{S}_h) \propto p(\boldsymbol{\phi}|\boldsymbol{S}_1, \dots, \boldsymbol{S}_{h-1}) \cdot p(\boldsymbol{S}_h|\boldsymbol{\phi})$$

Equation 2:13 – Naïve Bayesian approach for posterior computing [Broderick et al. 2013].

This approach is equivalent to those in [Sato 2001; Koshinaka et al. 2009; Cournapeau et al. 2010] if the VBEM is iterated only once per sequence S_h instead of waiting until convergence.

In [Lin 2013], the on-line elaboration enriches the VBEM with a nonparametric behavior: instead of initializing the number of HMM states to N or the number of Gaussian components to K, the proposed algorithm begins with an empty model ($N = K = 1$) and progressively refines it as data come in, adding new states and/or new components when needed. Therefore, the model complexity is adapted on-the-fly through a sequential approximation of the posterior based on the predictive distribution.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Matteo Causo, Lille 1, 2017

2.2.5 Distribution

Portable, wearable and implantable electronics is generally constrained to a meager power and memory budget and requires intelligence to be as close as possible to data sources. Intuitively, local elaboration of small amounts of data, further simplifies data structures and local computation at the expense of an increased communication overhead [Attiya and Welch 2004].

In the state-of-the-art VBEM solution introduced in Section 2.2.2, the whole computation is confined to a single powerful machine in a centralized manner therefore the processing is ex-situ, i.e. far away from the data source. In order to bring the elaboration in-situ and to reduce its complexity, distributed batch solutions have recently been proposed.

In [Broderick et al. 2013], a temporal distribution addresses fast input streams. In fact, a powerful machine is in charge of partitioning the input data in consecutive sequences called mini-batches and sends them to several nodes. Each of them computes the local posterior and sends it back to the powerful machine acting as a centralized fusion node. The fusion of the posterior is however obtained by a naïve exploitation of the Bayes' rule in Equation 2:13, which leads to poor approximations for unsupervised models with inherent symmetry, due to the approximation used in Variational inference. In [Campbell and How 2014] this issue is addressed by including an additional optimization step in the combination procedure that accounts for these broken dependencies and it does not require a centralized fusion node. Nevertheless this additional optimization step involves parameter permutation symmetry leading to a computational overhead and more approximation than the Variational framework would require. Finally, an interesting spatial distribution has been proposed very recently in [Hua and Li 2016], where information to merge is obtained from one-hop neighbors to target decentralized information processing and inference tasks over arbitrary and with limited communication capabilities Wireless Sensor Networks (WSNs). Although generally valid for the Variational Bayesian framework, it has been only applied to the GMM and its performance is almost comparable to the centralized solution.

2.2.6 Semi-supervision

According to [Russell and Norvig 2013], there are three broad categories of Machine Learning (ML) tasks:

> *Supervised learning*: The learning system is presented with example inputs and their desired outputs. The goal is to learn a general rule that maps inputs to outputs.
>
> *Unsupervised learning*: The learning system is presented with example inputs but no labels for them. The learning system is left alone to find similarities in the inputs. The goal is both to discover hidden patterns and to learn invariant features.
>
> *Reinforcement learning*: The learning system is presented with nothing. It interacts with a dynamic environment that awards or punishes it according to the learning objective.

When labeled training is plentiful, supervised learning techniques are widely used since they give excellent generalization performance [Bishop and Lasserre 2007]. However, Big Data come generally unlabeled, so unsupervised learning seems to be the most appropriate approach for them. Nevertheless, without any supervision model misspecifications could occur, i.e. the learnt model is different from the true distribution generating the data. In practice, the generalization performance of unsupervised learnt models is often poorer than of supervised learnt ones. In the attempt to gain the benefit of both supervised and unsupervised learning approaches, heuristic procedures have been proposed [Bouchard and Triggs 2004; Holub and Perona 2005; Yakhnenko et al. 2005] to interpolate between these two extremes as hybrid approaches referred to as *semi-supervised learning*, where the learning system is presented with example inputs and incomplete labels.

In [Bishop and Lasserre 2007], semi-supervised learning is interestingly reinterpreted in statistical terms[11] as the *discriminative training of a generative model*. Authors illustrate the advantages of blending between generative and discriminative viewpoints because a discriminative model cannot make use of unlabeled data, while a generative model could result with some model misspecifications and needs to be guided towards the true model of the reality. They propose a new perspective in which there is only one correct way to train a given model and a discriminatively trained generative model is fundamentally a new model because generative and discriminative models correspond to specific choices for the prior over parameters.

[11] The correspondence between ML and statistics jargon [Wu et al. 2009; Bishop and Lasserre 2007] defines a discriminative model as supervised learning a probabilistic model, while a generative model as unsupervised learning a probabilistic model and the process that generates the data.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

The treasure hunt example could help gaining a better understanding. We strive to get a treasure that is the knowledge we would like to extract from data. If the map is given and so are the solutions to all tests, the model we are exploiting is a discriminative one, because the machine has already the knowledge about where the treasure is and how to get it, i.e. the Maximum Likelihood estimate situation. Therefore, it has only the task to get to the treasure given the input data. Instead, in the case only the map is given, but no clues, or neither the map nor the clues are given, the machine needs to compensate to this lack of knowledge by guessing what the map and the clues may look like and then go looking for the treasure. This is exploiting a generative model that represents the most common case in practice, where some hidden information, i.e. data labels, has to be figured out first. Between these two extremes, if part of the map and some of the clues are given, it would result intuitively easier to complete the information set and learn the true model, i.e. hidden information is constrained to known one depending on the prior choice.

2.3 Conclusions

The background on Bayesian Machine Learning (BML) and the state-of-the-art analysis provided in this Chapter support our motivation in considering the BML as a valid solution for replicating the human brain working principles. The main idea behind this PhD thesis discussed in Section 1.5 is here revised to reflect of the acquired concepts, Figure 2:11.

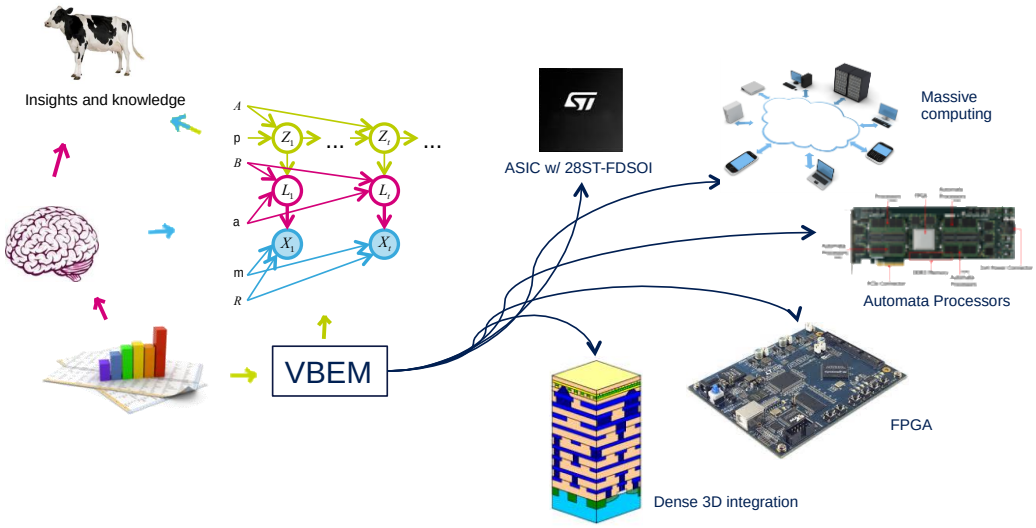


Figure 2:11 Revised illustration of our idea.

(The human brain retrieves insights and knowledge from raw or preprocessed inputs (fuchsia arrows). It however employs its plastic structure to process information and model the world (light-blue arrows). Our idea is to exploit Variational Bayesian Expectation-Maximization (VBEM) to model the world with a HMM GMM and to retrieve insights and knowledge from input data (light-green arrows). This sought behavior is independent from its actual implementation, so the VBEM can be mapped in several HW technologies (dark-blue arrows))

BML models the reality on the basis of input data and employs the resulting model for prediction on future trends or high-level elaboration. It is not yet known if the human brain behaves in such way, but the HTM principles and theory of Section 1.3.4 are well explained by the Bayesian framework. Furthermore, advanced BML algorithms like the VBEM are able to learn the model structure on-the-fly, resembling the brain ability to plastically shaping its structure. The two mechanisms may be different. In fact, the BML is selecting the right model among a set of possible ones, while the human brain may have the capability to build the model itself. Anyhow, homogeneity in the neo-cortex structure suggests the brain performs the same algorithm everywhere so that the model it employs may be generic to cope with a number of different applications. According to the main objective of this PhD thesis work, we mainly target IoT and medical applications generally related to stream analytics. For this reason, we have chosen a HMM GMM as generic structure. The target application, however, results independent on the actual implementation because the model BML creates, is implicit and populates the probabilistic space, i.e. it is not real. BML algorithms and their execution instead are real and demand for optimization in order to tackle Big Data challenges.

The state-of-the-art VBEM algorithm in [Ji et al. 2006] has been introduced as an interesting learning solution for HMM GMM parameters and structure. However, its naïve HW implementation results neither effective nor feasible because of the required memory, computing power and overall complexity. Therefore, this PhD thesis has been devoted to simplify and enhance the VBEM HMM GMM algorithm, so that its HW implementation becomes both feasible and attractive. In particular, we propose an on-line, distributed and nonparametric version of the state-of-the-art VBEM HMM GMM. We take advantage of the statistical perspective to

© 2017 Tous droits réservés.

Thèse de Matteo Causo, Lille 1, 2017

semi-supervision discussed in Section 2.2.6, which allows us to move the focus from labeled or unlabeled input data to knowing or not knowing the probabilistic model structure. Therefore, the supervised component of the semi-supervised learning relies on providing the machine with a prior on the probabilistic model structure, so that the unsupervised component is focusing on estimating the model parameters, i.e. the knowledge to extract.

The paramount principles behind HTM theory, namely continuous prediction; inference; sequential memory; closed-loop learning by considering time as supervisor are therefore widely exploited.

The proposed algorithm can be mapped to Spiking Neural Networks by conveniently reversing the work in [Nessler et al. 2013; Deneve 2015]. This way a conscientious estimation on the number of spiking neurons and their connections can be defined. Otherwise, a Deep Learning implementation with a conscious number of hidden layers can be obtained by reversing the work of [Mallat 2016; Gal and Ghahramani 2015b].

Our solution is independent from its actual HW implementation. In fact, the proposed algorithm can be mapped to a conventional HW where further optimization techniques can be applied to automata processors and many other structures that accomplish the Bayesian framework. We demonstrate this point by designing a proof-of-concept ULP HW implantation on FPGA platform for real-time streaming application.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Matteo Causo, Lille 1, 2017

References

M.J. Alam, P. Kenny, P. Dumouchel, and D. O'Shaughnessy. 2014. Noise spectrum estimation using Gaussian mixture model-based speech presence probability for robust speech recognition. In *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*. 2759–2763.

David J. Aldous. 1983. Exchangeability and related topics. *École d'Été Probab. Saint-Flour XIII — 1983* (1983), 1–198. DOI:http://dx.doi.org/10.1007/BFb0099421

Hagit Attiya and Jennifer Welch. 2004. Distributed Computing. (2004), 414. DOI:http://dx.doi.org/10.1002/0471478210

M.A. Balafar. 2012. Gaussian Mixture Model Based Segmentation Methods for Brain MRI Images. *Artif. Intell. Rev.* (2012), 429–439. DOI:http://dx.doi.org/10.1007/s10462-012-9317-3

David Barber. 2011. Bayesian Reasoning and Machine Learning. *Mach. Learn.* (2011), 646. DOI:http://dx.doi.org/10.1017/CBO9780511804779

Matthew J. Beal, Zoubin Ghahramani, and Carl E. Rasmussen. 2002. The Infinite Hidden Markov Model. In *Machine Learning*. 29–245. DOI:http://dx.doi.org/10.1.1.16.2929

MJ Matthew J. Beal. 2003. Variational algorithms for approximate Bayesian inference. *PhD Thesis* , May (2003), 1–281.

C.M. Bishop and Julia Lasserre. 2007. Generative or discriminative? getting the best of both worlds. *Bayesian Stat.* 8 (2007), 3–24.

David Blackwell and James B. MacQueen. 1973. Ferguson Distributions via Polya Urn Schemes. *Ann. Stat.* 1, 2 (1973), 353–355. DOI:http://dx.doi.org/10.1214/aos/1176342372

Guillaume Bouchard and William Triggs. 2004. The trade-off between generative and discriminative classifiers. *16th IASC Int. Symp. Comput. Stat.* (2004), 721–728.

Stephen Boyd and Lieven Vandenberghe. 2004. *Convex Optimization*, Cambridge: Cambridge University Press.

Tamara Broderick, Nicholas Boyd, Andre Wibisono, Ashia C. Wilson, and Michael I. Jordan. 2013. Streaming Variational Bayes. In *NIPS*. 1–9.

Wray L. Buntine. 1994. Operations for Learning with Graphical Models. *J. Artif. Intell. Res.* 2 (1994), 159–225. DOI:http://dx.doi.org/10.1613/jair.62

Trevor Campbell and Jonathan P. How. 2014. Approximate Decentralized Bayesian Inference. *Nips* (2014).

M.Á. Carreira-Perpiñán. 2000. Mode-finding for mixtures of gaussian distributions. *IEEE Trans. Pattern Anal. Mach. Intell.* 22, 11 (2000), 1318–1323. DOI:http://dx.doi.org/10.1109/34.888716

D. Cournapeau, S. Watanabe, a Nakamura, and T. Kawahara. 2010. Online Unsupervised Classification With Model Comparison in the Variational Bayes Framework for Voice Activity Detection. *Sel. Top. Signal Process. IEEE J.* 4, 6 (2010), 1071–1083.

Sophie Deneve. 2015. Bayesian Inference with Spiking Neurons. In *Encyclopedia of Computational Neuroscience*. 361–364. DOI:http://dx.doi.org/10.1007/978-1-4614-6675-8_568

Chuong B. Do and Serafim Batzoglou. 2008. What is the expectation maximization algorithm? *Nat. Biotechnol.* 26, 8 (August 2008), 897–9. DOI:http://dx.doi.org/10.1038/nbt1406

S. Esmaeili, B.N. Araabi, H. Soltanian-Zadeh, and L. Schwabe. 2014. Variational Bayesian learning for Gaussian mixture HMM in seizure prediction based on long term EEG of epileptic rats. In *2014 21th Iranian Conference on Biomedical Engineering (ICBME)*. IEEE, 138–143. DOI:http://dx.doi.org/10.1109/ICBME.2014.7043909

Yarin Gal and Zoubin Ghahramani. 2015a. Bayesian Convolutional Neural Networks with Bernoulli Approximate Variational Inference. (2015), 12. DOI:http://dx.doi.org/10.1029/2003JD002581

Yarin Gal and Zoubin Ghahramani. 2015b. On Modern Deep Learning and Variational Inference. *Adv. Approx. Bayesian Inference Work. NIPS* (2015), 1–9.

Z. Ghahramani. 2001. An introduction to hidden Markov models and Bayesian networks. *Int. J. Pattern Recognit. …* 15, 1 (2001), 9–

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Matteo Causo, Lille 1, 2017

42.

Zoubin Ghahramani. 2013. Bayesian non-parametrics and the probabilistic approach to modelling. *Philos. Trans. A. Math. Phys. Eng. Sci.* 371, 1984 (2013), 20110553. DOI:http://dx.doi.org/10.1098/rsta.2011.0553

Zoubin Ghahramani. 2002. Graphical models : parameter learning. *Handb. Brain Theory Neural Networks* , 2 (2002), 486–490. DOI:http://dx.doi.org/10.1561/2200000001

Zoubin Ghahramani. 2015. Probabilistic machine learning and artificial intelligence. *Nature* 521, 7553 (2015), 452–459. DOI:http://dx.doi.org/10.1038/nature14541

Zoubin Ghahramani and Matthew J. Beal. 2001. Graphical Models and Variational Methods. *Adv. Mean F. methods - Theory Pract.* (2001), 161–178.

HALO. 2015. IEEE / ACM Workshop on Hardware and Algorithms for Learning On-a-chip (HALO) Summary Report. (2015).

Alex Holub and Pietro Perona. 2005. A discriminative framework for modelling object classes. In *Proceedings - 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005*. 664–671. DOI:http://dx.doi.org/10.1109/CVPR.2005.25

Junhao Hua and Chunguang Li. 2016. Distributed Variational Bayesian Algorithms Over Sensor Networks. *IEEE Trans. Signal Process.* 64, 3 (2016), 783–798. DOI:http://dx.doi.org/10.1109/TSP.2015.2493979

Shihao Ji, Balaji Krishnapuram, and Lawrence Carin. 2006. Variational Bayes for continuous hidden Markov models and its application to active learning. *IEEE Trans. Pattern Anal. Mach. Intell.* 28, 4 (2006), 522–532. DOI:http://dx.doi.org/10.1109/TPAMI.2006.85

M. Jordan. 2003. An introduction to probabilistic graphical models. *Univ. Calif.* 6 (2003), 37–44.

Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul. 1999. Introduction to variational methods for graphical models. *Mach. Learn.* 37, 2 (1999), 183–233. DOI:http://dx.doi.org/10.1023/A:1007665907178

Michael I. Jordan and Yair Weiss. 1996. Probabilistic inference in graphical models. *Lauritzen, S. L* 16, 510 (1996), 140–152.

Wael Khreich, Eric Granger, Ali Miri, and Robert Sabourin. 2010. On the memory complexity of the forward-backward algorithm. *Pattern Recognit. Lett.* 31, 2 (2010), 91–99. DOI:http://dx.doi.org/10.1016/j.patrec.2009.09.023

T. Koshinaka, K. Nagatomo, and K. Shinoda. 2009. Online speaker clustering using incremental learning of an ergodic hidden Markov model. *Proc. Int. Conf. Acoust. Speech, Signal Process.* (2009), 4093–4096. DOI:http://dx.doi.org/10.1109/ICASSP.2009.4960528

Dahua Lin. 2013. Online Learning of Nonparametric Mixture Models via Sequential Variational Approximation. *Adv. Nueral Inf. Process. Syst. 26 (Proceedings NIPS)* (2013), 1–9.

D.J.C. MacKay. 1997. Ensemble learning for hidden Markov models. *Tech. Rep.* , 1995 (1997).

John Maindonald. 2007. Pattern Recognition and Machine Learning Springer, ed. *J. Electron. Imaging* 16, 4 (January 2007), 49901. DOI:http://dx.doi.org/10.1117/1.2819119

Stéphane Mallat. 2016. Understanding Deep Convolutional Networks. *arXiv* (2016), 1–17. DOI:http://dx.doi.org/10.1063/1.4939230

Matthew J. Martin, Amanda M. Smelser, and George Holzwarth. 2016. Dividing organelle tracks into Brownian and motor-driven intervals by variational maximization of the Bayesian evidence. *Eur. Biophys. J.* 45, 3 (2016), 269–277. DOI:http://dx.doi.org/10.1007/s00249-015-1091-0

Geoffrey J. McLachlan and Thriyambakam Krishnan. 2008. *The EM Algorithm and Extensions, 2E*, Hoboken, NJ, USA: John Wiley & Sons, Inc.

F. Mekhalfa, N. Nacereddine, and A.B. Goumeidane. 2007. Unsupervised Algorithm for Radiographic Image Segmentation Based on the Gaussian Mixture Model. *EUROCON, 2007. Int. Conf. ``Computer as a Tool''* , November 2015 (2007), 289–293. DOI:http://dx.doi.org/10.1109/EURCON.2007.4400401

Petr Motlicek, Subhadeep Dey, Srikanth Madikeri, and Lukas Burget. 2015. Employment of Subspace Gaussian Mixture Models in speaker recognition. In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*. 4445–4449. DOI:http://dx.doi.org/10.1109/ICASSP.2015.7178811

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Matteo Causo, Lille 1, 2017

Radford M. Neal. 1996. *Bayesian Learning for Neural Networks*, New York, NY: Springer New York.

Bernhard Nessler, Michael Pfeiffer, Lars Buesing, and Wolfgang Maass. 2013. Bayesian Computation Emerges in Generic Cortical Microcircuits through Spike-Timing-Dependent Plasticity. *PLoS Comput. Biol.* 9, 4 (April 2013), e1003037. DOI:http://dx.doi.org/10.1371/journal.pcbi.1003037

T.M. Nguyen and Q.M.J. Wu. 2013. Fast and Robust Spatially Constrained Gaussian Mixture Model for Image Segmentation. *IEEE Trans. Circuits Syst. Video Technol.* 23, 4 (2013), 621–635. DOI:http://dx.doi.org/10.1109/TCSVT.2012.2211176

S. Okasha. 2002. Philosophical Theories of Probability. *Br. J. Philos. Sci.* 53, 1 (March 2002), 151–156. DOI:http://dx.doi.org/10.1093/bjps/53.1.151

Peter Orbanz and Yee Whye Teh. 2005. Bayesian Nonparametric Models. *Encycl. Mach. Learn.* 25, 1 (2005), 1–14. DOI:http://dx.doi.org/10.1016/S0169-7161(05)25010-1

Gene Ott. 1967. Compact Encoding of Stationary Markov Sources. *IEEE Trans. Inf. Theory* 13, 1 (1967), 82–86. DOI:http://dx.doi.org/10.1109/TIT.1967.1053960

Judea Pearl. 1988. Probabilistic Reasoning in Intelligent Systems. *Morgan Kauffmann San Mateo* 88 (1988), 552. DOI:http://dx.doi.org/10.2307/2026705

L.R. Rabiner. 1989. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proc. IEEE* 77, 2 (1989), 257–286. DOI:http://dx.doi.org/10.1109/5.18626

C.E. Rasmussen and Z. Ghahramani. 2001. Occam's razor. *History* (2001), 1–26.

Douglas a Reynolds. 2008. Gaussian Mixture Models. *Encycl. Biometric Recognit.* 31, 2 (2008), 1047–64. DOI:http://dx.doi.org/10.1088/0967-3334/31/7/013

S. Roweis and Z. Ghahramani. 1999. A unifying review of linear gaussian models. *Neural Comput.* 11, 2 (1999), 305–45. DOI:http://dx.doi.org/10.1162/089976699300016674

Stuart Russell and Peter Norvig. 2013. *Artificial Intelligence A Modern Approach*,

Masa-aki Sato. 2001. Online Model Selection Based on the Variational Bayes. *Neural Comput.* 13, 7 (2001), 1649–1681. DOI:http://dx.doi.org/10.1162/089976601750265045

Jürgen Schmidhuber. 2015. Deep Learning in neural networks: An overview. *Neural Networks* 61 (2015), 85–117.

J. Sethuraman. 1994. A constructive definition of Dirichlet priors. *Stat. Sin.* 4 (1994), 639–650. DOI:http://dx.doi.org/***

Stephen H. Shum, Najim Dehak, Reda Dehak, and James R. Glass. 2013. Unsupervised methods for speaker diarization: An integrated and iterative approach. *IEEE Trans. Audio, Speech Lang. Process.* 21, 10 (2013), 2015–2028.

Y.W. Teh and M.I. Jordan. 2010. Hierarchical Bayesian nonparametric models with applications. *Bayesian nonparametrics* (2010).

Yee Whye Teh, Michael I. Jordan, Matthew J. Beal, and David M. Blei. 2006. Hierarchical Dirichlet Processes. *J. Am. Stat. Assoc.* 101, 476 (2006), 1566–1581. DOI:http://dx.doi.org/10.1198/016214506000000302

Sue E. Tranter and Douglas A. Reynolds. 2006. An overview of automatic speaker diarization systems. *IEEE Trans. Audio, Speech Lang. Process.* 14, 5 (2006), 1557–1565. DOI:http://dx.doi.org/10.1109/TASL.2006.878256

Dimitris G. Tzikas, Aristidis C. Likas, and Nikolaos P. Galatsanos. 2008. The Variational Approximation for Bayesian Inference. *IEEE Signal Process. Mag.* , November (2008), 131–146.

Fabio Valente and Christian Wellekens. 2005. Variational bayesian adaptation for speaker clustering. *ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc.* I, 3 (2005). DOI:http://dx.doi.org/10.1109/ICASSP.2005.1415276

S. Watanabe, a. Sako, and a. Nakamura. 2006. Automatic determination of acoustic model topology using variational Bayesian estimation and clustering for large vocabulary continuous speech recognition. *IEEE Trans. Audio, Speech Lang. Process.* 14, 3 (2006), 855–872. DOI:http://dx.doi.org/10.1109/TSA.2005.857791

Jiang Wu, Yuan-Bo Diao, Meng-Long Li, Ya-Ping Fang, and Dai-Chuan Ma. 2009. A semi-supervised learning based method: Laplacian support vector machine used in diabetes disease diagnosis. *Interdiscip. Sci. Comput. Life Sci.* 1, 2 (June 2009), 151–155.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Matteo Causo, Lille 1, 2017

Oksana Yakhnenko, Adrian Silvescu, and Vasant Honavar. 2005. Discriminatively trained markov model for sequence classification. In *Proceedings - IEEE International Conference on Data Mining, ICDM*. 498–505. DOI:http://dx.doi.org/10.1109/ICDM.2005.52

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Chapter 3 Algorithm level optimizations

In this Chapter, we present the limitations of the state-of-the-art VBEM HMM GMM algorithm that make unfeasible its HW implementation, namely redundant information on data structures and dependence on infinite classification history. After a complexity analysis, we propose and adopt two important optimizations aimed at simplifying and enhancing the on-line VBEM HMM GMM. Therefore, we make its HW implementation both feasible and attractive.

3.1 On-line VBEM HMM GMM

The state-of-the-art VBEM HMM GMM in Section 2.2.2 is an iterative approach incrementally improving the learning and the consecutive classification accuracy until convergence is reached – that is, either the log-likelihood or the parameters estimation does not improve anymore beyond an acceptable tolerance. In the batch formulation, it needs to access the whole dataset at least once per iteration. Thus, data have to be stored and maintained in a big memory. From hardware prospective, this translates into an off-chip memory with consequent performance degradation due to the memory wall. This issue is solved in the on-line formulation, where learning and classification are incrementally improved by considering a different sequence $\boldsymbol{S}_h$ of input data every time. Consequently a smaller buffer memory is required and this structure is suitable for on-the-fly elaboration of continuous streams. Furthermore, we aim at replicating the human brain working principles according to the HTM theory, which actually relies both on on-line learning and continuous prediction.

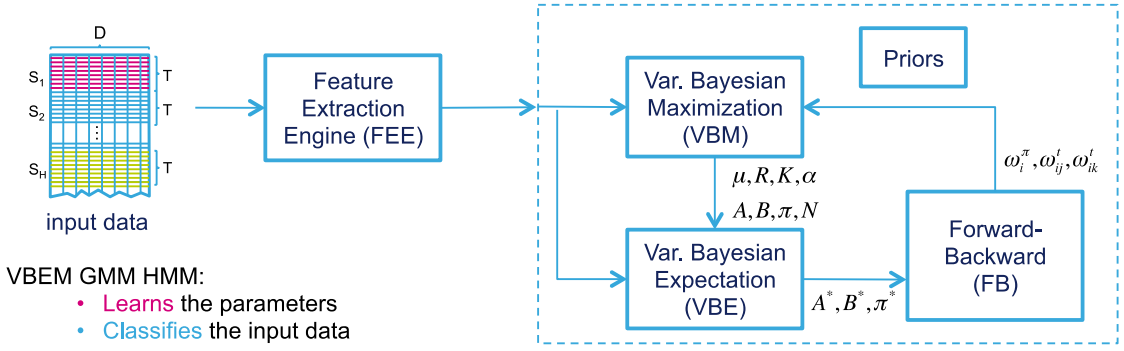


Figure 3:1 Block diagram of on-line VBEM HMM GMM.

Our approach to the on-line formulation for the VBEM HMM GMM follows the naïve Bayesian in [Broderick et al. 2013] discussed in Section 2.2.4. However, this approach establishes a temporal distribution among several nodes to speed up elaboration of continuous streams rather than building on a proper on-line formulation. In fact, every node is asynchronously assigned with a task to perform on local data by a centralized powerful node that collects and merges all partial results. In this way, high-performance design constrains are relaxed while the system is still able to perform fast stream analytics. Our idea is to exploit the naïve Bayesian fusion formula at the single powerful node to perform the on-line elaboration by incrementally refining the overall posterior calculation. In particular, the current posterior is employed as a prior for the next posterior calculation according to Equation 2:13. This is shown in Figure 3:1, where each sequence $\boldsymbol{S}_h$ of T consecutive input data is preprocessed by a Feature Extraction Engine (FEE) and presented to the VBEM HMM GMM. This latter is similar to the state-of-the-art in Figure 2:10 but it is iterated only once per sequence presentation.

3.2 Complexity analysis

The paramount objective of this PhD thesis work is to target IoT and medical applications with an Ultra-Low-Power (ULP) prospective, therefore we aim to simplify and enhance the on-line VBEM HMM GMM by means of optimizations without loss of generality. In fact, the on-line VBEM HMM GMM algorithm in Figure 3:1 has several weaknesses that slow down its execution and make its HW implementation unfeasible or not attractive for portable, wearable and implantable electronics.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

The Variational Bayesian Maximization (VBM) step learns the model parameters $(\boldsymbol{\mu},\boldsymbol{R},\boldsymbol{\alpha},\boldsymbol{A},\boldsymbol{B},\boldsymbol{\pi})$ and hyper-parameters (K,N) on the basis of the current input data and the sufficient statistics $(\omega_i^{\pi},\omega_i^h,\omega_{ij}^h)$ computed during the Forward-Backward (FB). This latter, in particular, performs inference on the complete information set $\boldsymbol{Y}$ obtained during the Variational Bayesian Expectation (VBE) and classifies input data either through the FB or the Forward Filtering Backward Smoothing (FFBS) algorithm. Both the VBE and the VBM need to see the input data and are performed only once per sequence in the on-line formulation of Figure 3:1. Conversely, the FB block performs inference on the whole classification history that however grows indefinitely as input data are continuously presented to the machine.

3.2.1 Forward-Backward (FB)

The FB algorithm is a dynamic programming technique that efficiently evaluates the likelihood $p(\boldsymbol{S}|\boldsymbol{\phi})$ and computes the sufficient statistics $(\omega_i^{\pi},\omega_i^h,\omega_{ij}^h)$, i.e. the smoothed conditional state probability densities for updating HMM parameters. It consists of the forward pass in Figure 3:2Left that computes $\alpha_h(i) \equiv p(Z_h = i, \boldsymbol{S}_{1:h}|\boldsymbol{\phi})$ and the backward pass in Figure 3:2Right that computes $\beta_h(i) \equiv p(\boldsymbol{S}_{h+1:H}|Z_h = i, \boldsymbol{\phi})$.

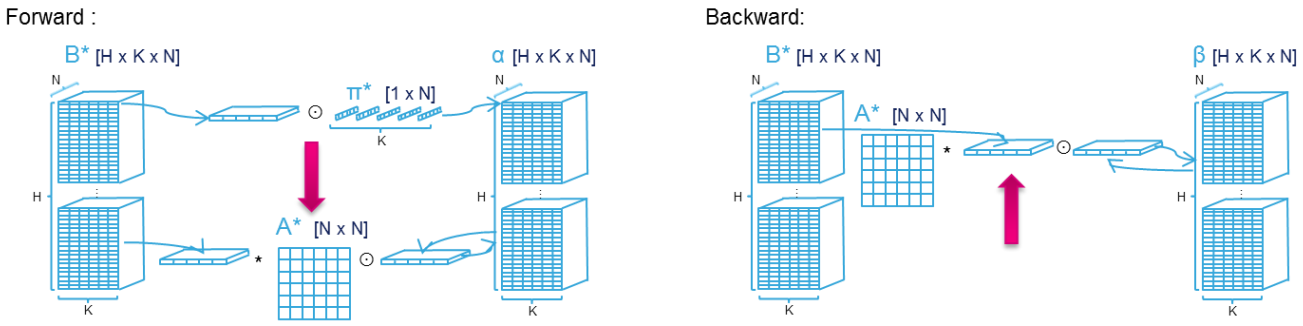


Figure 3:2 Schematic illustrations for the Forward (Left) and the Backward (Right) passes.

(This illustration gives an intuition of the complexity for the FB algorithm. Data structures are represented with their dimensionalities and required computations are schematized. In particular, fuchsia arrows indicate the direction for each recursive procedure. Finally, several matrix multiplications are required to perform overall inference with a memory complexity of $C_{Mem} = O(HN)$ and an execution complexity of $C_{Exe} = O(2HN^2)$.)

During the forward pass, each $[KxN]$ slice of $\boldsymbol{B}^*$ is multiplied by the state transition matrix $\boldsymbol{A}^*$and successively weighted with the previously computed $[KxN]$ slice of $\boldsymbol{\alpha}$, by means of the Hadamard product, i.e. element-wise multiplication represented with the symbol $\odot$. To initialize this recursive procedure, the first element of $\boldsymbol{\alpha}$ is conveniently computed by means of the initial state probability vector $\boldsymbol{\pi}^*$. Similarly, the backward pass requires initializing the first element of $\boldsymbol{\beta}$ to the all-ones $[KxN]$ matrix. In this case, however, the first element is the H-th because the backward direction goes from the most recent to the oldest classification event. Finally, sufficient statistics are computed according to Equation 3:1 where normalization ensures reasonable results and convergence.

$$\omega_i^{\pi} = \gamma_{(1|H)}(i) = p(\boldsymbol{Z}_1 = i|\boldsymbol{S}_{1:H}, \boldsymbol{\phi})$$

$$\omega_i^h = \gamma_{(h|H)}(i) = p(\boldsymbol{Z}_h = i|\boldsymbol{S}_{1:H}, \boldsymbol{\phi}) = \frac{\alpha_h(i) \cdot \beta_h(i)}{\sum_{n=1}^{N} \alpha_h(n) \cdot \beta_h(n)}$$

$$\omega_{ij}^h = \xi_{(h|H)}(i,j) = p(\boldsymbol{Z}_h = i, \boldsymbol{Z}_{h+1} = j|\boldsymbol{S}_{1:H}, \boldsymbol{\phi}) = \frac{\alpha_h(i) \cdot A_{ij} \cdot B_{hj} \cdot \beta_{h+1}(j)}{\sum_{n=1}^{N}\sum_{l=1}^{N} \alpha_h(n) \cdot A_{nl} \cdot B_{hl} \cdot \beta_{h+1}(l)}$$

Equation 3:1 – Forward Backward (FB) sufficient statistics.

Although Figure 3:2 shows the FB requires a big number of matrix multiplications, it is very efficient compared to the analytical maximization of the posterior because it exploits inference through message passing algorithms. However in order to obtain a feasible HW implementation, this block should be optimized because its memory complexity is estimated at $C_{Mem} = O(HN)$ and its execution complexity at $C_{Exe} = O(2HN^2)$. Indeed, the FB retraces the entire classification history forward; optimizes the current classification on the basis of already seen data backward. In other words, its complexity depends linearly on H, but H goes to infinity in the frame of continuous streams analytics [Khreich et al. 2010]. In particular, in the on-line formulation a new element is progressively added

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Matteo Causo, Lille 1, 2017

in the classification history on-the-fly, i.e. $H \to H+1$ and $H \gg N$. Therefore, the matrices $\boldsymbol{B}^*$, $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ in Figure 3:2 together with the sufficient statistics ω_i^h and ω_{ij}^h in Equation 3:1 grow indefinitely in H.

3.2.2 Forward Filtering Backward Smoothing (FFBS)

Proposed in [Ott 1967], the FFBS is another approach similar to FB but more stable and meaningful because it directly propagates probability densities [Khreich et al. 2012]. During the forward recursion, it computes the predictive density $\gamma_{(h|h-1)}(i) \equiv p(Z_h = i|\boldsymbol{S}_{1:h-1}, \boldsymbol{\phi})$ and the filtered density $\gamma_{(h|h)}(i) \equiv p(Z_h = i|\boldsymbol{S}_{1:h}, \boldsymbol{\phi})$, according to the matrix formulation in Equation 3:2.

$$\begin{aligned} \gamma_{(h|h-1)} &= \boldsymbol{A}^{*\prime}\gamma_{(h-1|h-1)} \\ \gamma_{(h|h)} &= \gamma_{(h|h-1)} \odot \boldsymbol{B}_h^* \oslash \mathbf{1}\gamma_{(h|h)} \end{aligned}$$

Equation 3:2 – Predictive and Filtered densities in FFBS.

where $\mathbf{1}$ is a row vector of ones; the prime superscript denotes the transpose operation; the symbol $\odot$ stands for the element-wise multiplication and $\oslash$ indicates the element-wise division. This is very interesting because it replicates the HTM principles. In fact, for each step of the forward pass, this algorithm performs inference through continuous prediction. In particular, it calculates the predictive probability by using the previous history encoded in the transition matrix $\boldsymbol{A}^*$ and then it tests its predictions with the actual data in $\boldsymbol{B}_h^*$ by resulting with the filtered probability. In this way, it checks the accuracy of the learned model up to the current time in a sort of closed-loop system as required in the HTM theory where time is supervisor. So, in this aspect the FFBS is much more interesting than the FB for the objectives of this PhD thesis. Furthermore, during the backward recursion the FFBS directly computes the required sufficient statistics referred with a b subscript in Equation 3:3 without accessing to $\boldsymbol{B}^*$. Indeed, it reuses the filtered probabilities computed during the forward pass, which it opportunely updates with the information gathered by spanning the entire classification history.

$$\begin{aligned} \omega_i^{\pi} &= \gamma_{(1|H)} \\ \omega_i^h &= \gamma_{(h|H)_b} = \xi_{(h|H)_b}\mathbf{1}' \\ \omega_{ij}^h &= \xi_{(h|H)_b} = \left[\gamma_{(h|h)} \odot \gamma_{(h+1|H)_b} \oslash \boldsymbol{A}^{*\prime}\gamma_{(h|h)}\right]\mathbf{1} \odot \boldsymbol{A}^* \end{aligned}$$

Equation 3:3 – Forward Filtering Backward Smoothing (FFBS) sufficient statistics.

Although the FFBS has the same order of complexity as the FB, it is actually slightly less complex [Khreich et al. 2010]. However the FB is historically more famous and used in practical applications.

3.2.3 Data structures

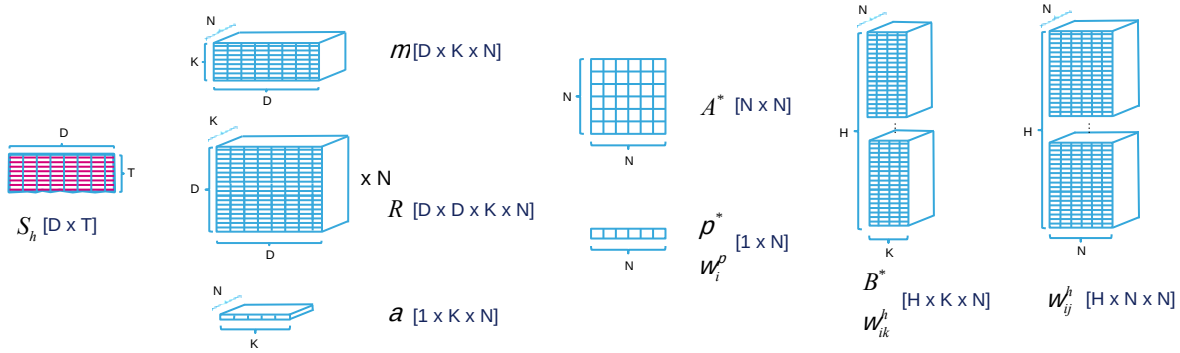


Figure 3:3 Complexity of online VBEM HMM GMM data structures.

In Figure 3:3, a closer view of VBEM HMM GMM shows the dimensionality of the internal data representation and highlights two principal problems, whose solutions are discussed below and identify some of the contributions of this PhD thesis work. Firstly, most parameters $(\boldsymbol{\mu},\boldsymbol{R},\boldsymbol{\alpha},\boldsymbol{B})$ and sufficient statistics (ω_{ik}^h, ω_{ij}^h) have both dimensionalities in K and in N in order to encode the notion of assigning each HMM state with a particular configuration of GMM components. This information is therefore redundant and implies a complexity in managing and elaborating these data. Secondly, $\boldsymbol{B}$, ω_{ik}^h and ω_{ij}^h depend linearly on the number of sequences H. In a

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

streaming application, H goes to infinity and makes unfeasible to map this architecture into a finite physical device. Therefore, exactly learning the overall model parameters and managing redundant information into data structures is neither optimal nor convenient for a HW implementation.

3.3 Optimization and complexity reduction

The structure we propose in Figure 3:4 addresses problems of the on-line VBEM HMM GMM by separating the algorithm in two parts. The first includes the VBE HMM GMM and the VBEM GMM, while the second the FB and the VBM HMM.

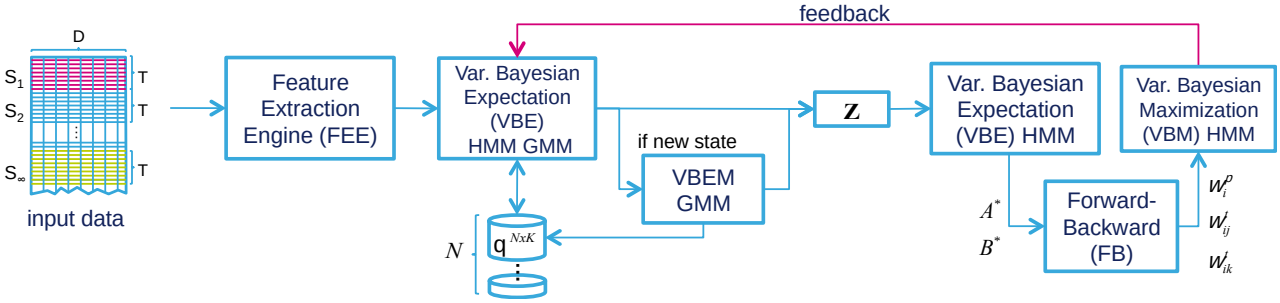


Figure 3:4 Block diagram of proposed on-line VBEM HMM GMM.

VBE HMM GMM and VBEM GMM interface input data; partially learn the model parameters and extract a sort of intermediate knowledge that is exploited by the FB and the VBM HMM to complete the model learning without seeing the input data. This separation allows for simpler data structures that depend either on K or N. In particular, input data are still arranged in sequences $\boldsymbol{S}$ and presented to the Bayesian classifier after the FEE extracted invariant features in a Gaussian-friendly arrangement to improve the final classification accuracy. The VBE HMM GMM computes the expectations the current sequence $\boldsymbol{S}_h$ has been generated by already learnt models. Thus, it results with the HMM state Z_n whose Gaussian components best explain input data. If none of the existing states is able to explain the input data, a new HMM state Z_{N+1} is introduced on-the-fly and its related Gaussian parameters are learnt by executing the VBEM GMM until convergence. Both the FB and the VBM HMM exploit this information on $\boldsymbol{Z}$ to compute the sufficient statistics and improve HMM parameters estimations. Consequently, the complexity of $\boldsymbol{B}$, the biggest data structure, and ω_{ik}^{h} reduces, in fact they both lose their dependence on K. Nevertheless, GMM parameters $\boldsymbol{\theta}$ keep their dimensionalities both in K and N. A further simplification consists in substituting the VBE HMM GMM with the deterministic assignment step of the k-means algorithm [Hartigan and Wong 1979], where each observation is assigned to the cluster whose mean $\boldsymbol{\mu}$ yields the least Within-Cluster Sum of Squares (WCSS). In this case, ($\boldsymbol{R}$,$\boldsymbol{\alpha}$) can also be reduced because only $\boldsymbol{\mu}$ is utilized and relates GMM parameters to the HMM state they belong to.

Mathematically, our approach requires Equation 2:9 to be reformulate as in Equation 3:4, with $\boldsymbol{\phi} = \{\boldsymbol{\pi}^N, \boldsymbol{A}^{NxN}, \boldsymbol{\theta}_h^{NxK}\}$ and $\boldsymbol{B}_h \equiv p(\boldsymbol{\theta}_h|\boldsymbol{Z}_h) \cdot \prod_t^T p(\boldsymbol{L}_t)p(\boldsymbol{X}_t|\boldsymbol{L}_t, \boldsymbol{\theta}_h)$, where $\boldsymbol{\theta}_h = \{\boldsymbol{\alpha}_h^K, \boldsymbol{\mu}_h^{NxKxD}, \boldsymbol{R}_h^{KxDxD}\}$.

$$p(\boldsymbol{Y}|\boldsymbol{\phi}) = \boldsymbol{\pi}\left[\prod_{h=2}^{H} \boldsymbol{A}_h\right]\prod_{h=1}^{H} \boldsymbol{B}_h$$

Equation 3:4 – Joint distribution for the HMM GMM in the on-line formulation.

Moreover, the VBEM HMM GMM in Figure 3:1 is required to iterate until convergence and HMM GMM parameters to continuously adapt on the basis of input data. Instead, our approach allows the VBE HMM GMM, the FB and the VBM HMM to be executed only once per sequence according to the on-line elaboration proposed in [Sato 2001]. In our case, however, HMM parameters estimation is incrementally improved and adapted to the input data, while GMM parameters do not change any longer after the VBEM GMM converged. Intuitively, GMM parameters identify a specific HMM state Z_n that generated the input sequence $\boldsymbol{S}_h$, the same GMM parameters were learnt on. For successive input sequences, the VBE HMM GMM identifies those sequences underlying the same cause, i.e. the sequences generated by the same HMM state Z_n, even if they are apparently and deterministically different. However, if input data are too different to imply this common cause, our approach introduces a new HMM state Z_{N+1}, which better explains the misclassified input data. This state will be eventually merged to the correct one during inference and HMM parameters learning performed respectively in the FB and the VBM HMM blocks.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Matteo Causo, Lille 1, 2017

3.3.1 Bayesian Nonparametric behavior

Another key point for the structure in Figure 3:4 concerns the VBEM capability to automatically infer the model complexity through Bayesian model selection. In order to prevent under-fitting in the Bayesian nonparametric framework, an infinite model complexity is assumed and it is reduced by the Bayesian approach, which is known to prevent over-fitting. Although this is a sought feature, it demands for a computational overhead to operate in an unbounded hidden space. Practically, the model or the variational distribution is truncated to a fixed model complexity, i.e. N and K. However, it still implies to overcharge the computation with inconvenient complexity that will affect performance in hardware acceleration.

Following the idea in [Lin 2013], we provided our method with a nonparametric behavior so that it adapts model complexity on-the-fly. Rather than randomly initializing a fixed number of states N, we begin with an empty model ($N = 1$) that is progressively refined with new input data by adding new states or pruning old ones on-the-fly when needed. The pruning is evaluated according to the effective utilization of the recognized states and it is triggered regularly during the learning process. Adding new states is regulated by the predictive probability distribution in Equation 3:5 efficiently computed by FB and VBM HMM that is closing the loop with the VBE HMM GMM. In particular, if the current model results in a small predictive probability, it poorly explains a sequence $\boldsymbol{S}_h$ and a new HMM state needs to be added. Only in this case, the VBEM GMM has to run until convergence in order to estimate the new mean $\boldsymbol{\mu}$ of Gaussian components. A further possible simplification may imply a nonparametric formulation of the VBEM GMM. However, VBEM GMM will be run sporadically in a batch mode and only on the data of the current sequence $\boldsymbol{S}_h$. Therefore, this further optimization would not provide any improvement or simplification of the algorithm with respect to the objective of making hardware acceleration feasible and attractive.

$$\begin{aligned} p(\boldsymbol{S}_h|\boldsymbol{S}_1,\dots,\boldsymbol{S}_{h-1}) &= \int p(\boldsymbol{S}_h|\boldsymbol{\phi}) \cdot p(\boldsymbol{\phi}|\boldsymbol{S}_1,\dots,\boldsymbol{S}_{h-1})\, d\boldsymbol{\phi} \\ &= \int \sum_{\boldsymbol{Z},\boldsymbol{L}} p(\boldsymbol{Y}_h|\boldsymbol{\phi}) \cdot p(\boldsymbol{\phi}|\boldsymbol{S}_1,\dots,\boldsymbol{S}_{h-1})\, d\boldsymbol{\phi} \end{aligned}$$

Equation 3:5 – Predictive probability distribution for the HMM GMM in the on-line formulation.

Finally, our solution in Figure 3:4 results less complex than the one in Figure 3:1 because it strongly benefits of the on-line non-parametric computation: N is incremented progressively from 1; data structures are simplified; VBE HMM, FB and VBM HMM do not need to see the data; VBE HMM GMM, VBE HMM, FB and VBM HMM require one iteration per sequence and the VBEM GMM runs until convergence only when a new HMM state is detected.

3.4 Mapping infinite history into finite resources

The on-line formulation discussed above is not yet exhaustive. For each new sequence S_h, the FB block in Figure 3:4 still performs complete passes on $\boldsymbol{B}^*$ for retracing the classification history with an overall time complexity of $O(N^2H^2)$. For a number of sequences H and a fully-connected HMM with N states, the memory complexity of both FB and FFBS algorithms grows linearly with H and N, $O(NH)$, and their time complexity grows quadratically with N and linearly with H, $O(N^2H)$. Generally, $H \gg N$, so the interest is in eliminating the dependence on H. This is particularly true for a continuous stream application, where the stream may be indefinitely long, and H can be assumed to go to infinity [Khreich et al. 2012]. In fact, the bigger the analyzed sequences number H, the larger the classification history and the more accurate the learning process. In practice, H has to be limited, otherwise it cannot be feasibly mapped to finite or, even worse, limited resources hardware as in the case of portable and wearable electronics. This is subject of ongoing research. In fact, the checkpoint algorithm [Grice et al. 1997] divides the input sequence into $\sqrt{H}$ sub-sequences rather than storing the whole matrix of filtered state densities. Its implementation reduces the memory complexity to $O\left(N\sqrt{H}\right)$, yet increases the time complexity to $O\left(N^2\left(2H-\sqrt{H}\right)\right)$. In [Sivaprakasam and Shanmugan 1995] the backward recursion is avoided for a further reduction in memory complexity to $O(N^2+N)$, while the time complexity augments for the forward pass to $O(N^4H)$.

We definitely eliminate the dependence on H by reformulating the FB block in Figure 3:4 to take advantage of the on-line elaboration. Both FB and FFBS algorithms require a forward pass and then a backward pass on the $\boldsymbol{B}^*$ matrix in order to compute the required sufficient statistics in Equation 3:3. The forward pass reduces to the incremental evaluation and update of previous statistics according to new observations, Equation 3:2. Therefore, we can take advantage of the on-line elaboration to simplify the forward pass. Secondly, the backward pass always starts with the quantities resulting from the forward pass that are correctly computed, Figure 3:5. Therefore, the first step of the backward pass is not approximated, but exact. The approximation arises because we do not retrace back the $\boldsymbol{B}^*$ matrix but incrementally update the desired backward sufficient statistics in a forward manner as well. This

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Matteo Causo, Lille 1, 2017

is the reason we named it the greedy forward pass. Exact sufficient statistics in Equation 3:3 can be easily compared with the approximated ones in Equation 3:6 computed during the forward pass, i.e. identified with a f subscript. Thus, the error is analytically evaluated in Equation 3:7.

$$\omega_i^h \cong \gamma_{(h-1|h)_f} = \xi_{(h-1|h)_f}\mathbf{1}'$$
$$\omega_{ij}^h \cong \xi_{(h-1|h)_f} = \left[\gamma_{(h-1|h-1)} \odot \gamma_{(h|h)_f} \oslash \boldsymbol{A}^{*\prime}\gamma_{(h-1|h-1)}\right]\mathbf{1} \odot \boldsymbol{A}^*$$

Equation 3:6 – Approximated sufficient statistics for the greedy forward pass.

$$\xi_{(h|H)_b} = \xi_{(h-1|h)_f} \odot [\textstyle\prod_{l=h+2}^{H} \varepsilon_l]\mathbf{1}$$
$$\gamma_{(h|H)_b} = \gamma_{(h-1|h)_f} \odot \textstyle\prod_{l=h+2}^{H} \varepsilon_l \quad \text{with} \quad \varepsilon_l = \boldsymbol{B}_l^* \odot \boldsymbol{A}^*\mathbf{1}' \oslash \mathbf{1}[\gamma_{(l|l-1)} \odot \boldsymbol{B}_l^*]$$

Equation 3:7 – Analytic comparison and error estimation of the greedy forward pass.

As reported in Figure 3:5, there is no approximation for last computed densities because $\xi_{(H-1|H)_b} = \xi_{(H-1|H)_f}$ and $\gamma_{(H-1|H)_b} = \gamma_{(H-1|H)_f}$. Along the overall forward pass, this implies that $\xi_{(h|H)_b} = \xi_{(h-1|h)_f}$ and $\gamma_{(h|H)_b} = \gamma_{(h-1|h)_f}$, i.e. we are building on a greedy forward pass that for every h is choosing the locally optimal HMM state for the current classification without seeing future classifications.

Following the idea in [Moore and Ford 1998; Digalakis 1999], we reformulate $\xi_{(h|H)_b}$, $\xi_{(h|h+1)_f}$, $\gamma_{(h|H)_b}$ and $\gamma_{(h|h+1)_f}$ as:

$$\xi_{(h|H)_{b_{ij}}} = \delta\left(\xi_{(h|H)_{b_{ij}}} - \max\left(\xi_{(h|H)_b}\right)\right) \quad ; \quad \xi_{(h-1|h)_{f_{ij}}} = \delta\left(\xi_{(h-1|h)_{f_{ij}}} - \max\left(\xi_{(h-1|h)_f}\right)\right)$$
$$\gamma_{(h|H)_{b_i}} = \delta\left(\gamma_{(h|H)_{b_i}} - \max\left(\gamma_{(h|H)_b}\right)\right) \quad\quad \gamma_{(h-1|h)_{f_i}} = \delta\left(\gamma_{(h-1|h)_{f_i}} - \max\left(\gamma_{(h-1|h)_f}\right)\right)$$

Equation 3:8 – Formulation following from [Digalakis 1999].

where $\delta\left((\cdot)_{ij} - max(\cdot)\right) = 1$ iff $(\cdot)_{ij} = max(\cdot)$, so that our greedy forward pass results with the Most Likely State Sequence (MLSS), which is the output of the Viterbi decoder. In this way, the state transition matrix $\boldsymbol{A}^*$ and the emission matrix $\boldsymbol{B}^*$ become frequency counters.

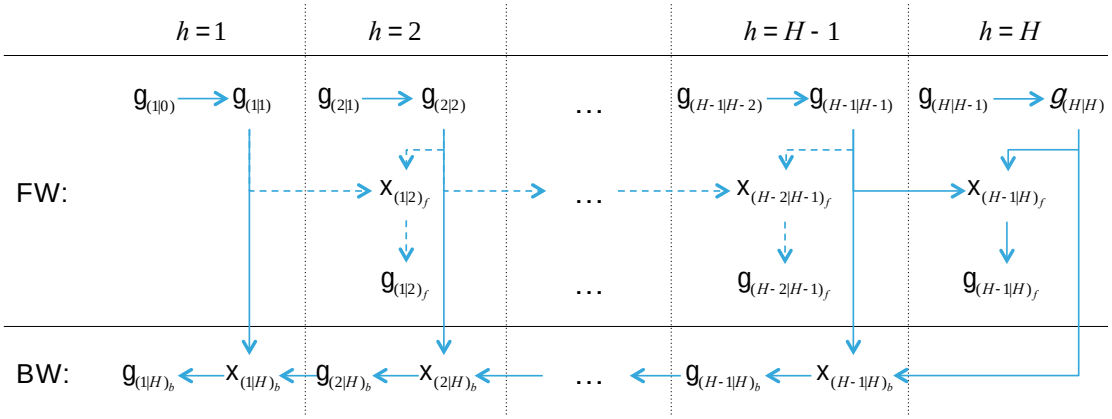


Figure 3:5 Forward (FW) and Backward (BW) pass for the batch FFBS.

(Solid arrows indicate conventional computation flow, while dash arrows the greedy forward recursion.)

The on-line formulation for the greedy forward pass easily follows. For each new sequence S_h, densities $\xi_{(h-1|h)_f}$ and $\gamma_{(h-1|h)_f}$ are directly derived from $\gamma_{(h-1|h-1)}$ and $\gamma_{(h|h)}$ without requiring any passes through the past classification history. This implies a time complexity per analyzed sequence reduced to $O(N^2)$ and a memory complexity of $O(N^2)$ because $\boldsymbol{B}$, ω_i^h and ω_{ij}^h in Equation 3:3 simplify and lose their dependence on H. Moreover, HMM parameters are estimated incrementally with $\boldsymbol{A}_h = \boldsymbol{A}_{h-1} + \xi_{(h-1|h)_f}$ and $\boldsymbol{B} = \gamma_{(h-1|h)_f}$. The parameter $\boldsymbol{\pi} = \gamma_{(1|2)_f}$ is assumed not to change over h.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Matteo Causo, Lille 1, 2017

The same complexity reduction was obtained for the FB in [Stenger et al. 2001; Florez-larrahondo et al. 2001] by approximating the backward pass quantities β_h respectively to 1 and to $A_h B_{h+1}$. Besides, the advantages FFBS presents over FB for its intrinsic stability and its slightly smaller complexity [Khreich et al. 2010], our approach results to be more suitable for a hardware implementation because of the formulation in Equation 3:8 and so treating HMM parameters as frequency counters, allows for further simplifications. For instance, Equation 3:2 reduces to Equation 3:9 where neither multiplications nor divisions are required. In fact, the element-wise product becomes an identity and the matrix multiplication translates in the selection of the row for which $\boldsymbol{B}$ is unitary, i.e. $\forall i \in N^+ | B_{h-1}(i) == 1$.

$$\gamma_{(h|h-1)} = \boldsymbol{A}'\boldsymbol{B}_{h-1} = \boldsymbol{A}'_{i,1:N}$$
$$\gamma_{(h|h)} = \boldsymbol{B}_h$$

Equation 3:9 – Predictive and Filtered densities for the greedy forward pass.

3.5 Overall on-line Algorithm

The pseudo-code of the overall algorithm for the simplified on-line VBEM HMM GMM is reported below. It is performing data-driven elaboration because triggered on new data arrival and data-centric elaboration because able to adapt the model structure and parameters to data. The algorithm starts unaware of its objective, but it is provided with the model parameters prior as required in the Bayesian framework. It learns in an unsupervised way what to do with input data and how to treat them. It gets an idea of what the underling probabilistic model looks like, and exploits it for prediction and future classifications. In particular, as depicted in Figure 3:4, the VBE HMM GMM computes the expectations that already learnt models generated the input sequence $\boldsymbol{S}_h$. The algorithm then exploits them to evaluate the predictive probability distribution in Equation 3:5, and if the threshold chosen according to [Dixit et al. 2011] is not overcome, a new GMM model $\boldsymbol{\theta}_{Z_{h,N+1}}$ has to be learnt through VBEM GMM to actually explain input data. The algorithm automatically decides at which sequence $\boldsymbol{S}_h$ to stop the learning with the same stop-learning criteria commonly exploited for the batch VBEM HMM GMM, i.e. stop iterating when the log-likelihood or the parameters estimation does not improve anymore beyond an acceptable tolerance. In our approach, however, the learning can be restarted considering the dynamic evolution of the system. In particular, when a new HMM state is detected by a small prediction probability, parameters are incrementally estimated until criteria for stopping learning are met again. Finally, VBE HMM GMM computes the expectations that already learnt HMM models would have resulted in $\boldsymbol{Z}$. The greedy forward pass results with the sufficient statistics the VBM HMM exploits to improve the HMM parameters estimation.

ALGORITHM 2. proposed on-line VBEM HMM GMM

Input: Sequence $\boldsymbol{S}_h = \{\boldsymbol{X}_{1:T,h}\}$ of input data.
Output: Classification result ω_i^h and HMM transition matrix $\boldsymbol{A}^*$.
Priors: α_0; β_0; $\boldsymbol{m}_0$; $\boldsymbol{R}_0$; u_{ij}^A; u_i^B

// VBE HMM GMM : [Bishop 2006 Eqs. 10.46-47; Ji et al. 2006 Eq. 44]
$\ln[\rho(\boldsymbol{X}_{t,h}|\boldsymbol{\theta}_{h,Z_h,L_t})] = \mathbb{E}[\ln(\boldsymbol{\alpha}_{h,Z_h,L_t})] - \frac{D}{2}\ln(2\pi) + \mathbb{E}[\ln|\boldsymbol{R}_{h,Z_h,L_t}|] - \frac{1}{2}[(\mathrm{X_{t,h}} - \mu_{h,Z_h,L_t})'\boldsymbol{R}_{h,Z_h,L_t}(\mathrm{X_{t,h}} - \mu_{h,Z_h,L_t})]$
$q^*(\boldsymbol{Z},\boldsymbol{L}) \propto \prod_{t=1}^{T}\prod_{n}^{N}\prod_{k}^{K}\left[\boldsymbol{A}^*\rho(\boldsymbol{X}_{t,h}|\boldsymbol{\theta}_{h,Z_{hn},L_{tk}})^{L_{tk}}\right]^{Z_{hn}}$

if $p(\boldsymbol{S}_h|\boldsymbol{S}_{1:h-1}) > 0.6$ [Dixit et al. 2011]
  $Z_i = \delta(\sum_{\boldsymbol{L}} q_i^*(\boldsymbol{Z},\boldsymbol{L}) - \max(\sum_{\boldsymbol{L}} q^*(\boldsymbol{Z},\boldsymbol{L})))$
else // new HMM state: VBEM GMM
  while GMM parameters change **do**
    $r_{tk} = \frac{\rho(\boldsymbol{X}_{t,h}|\boldsymbol{\theta}_{h,Z_h,L_t})}{\sum_{L_t}\rho(\boldsymbol{X}_{t,h}|\boldsymbol{\theta}_{h,Z_h,L_t})}$; $n_k = \sum_t^T r_{tk}$; $\bar{\boldsymbol{x}}_k = \frac{1}{n_k}\sum_t^T r_{tk}\boldsymbol{X}_{t,h}$
    $\boldsymbol{S}_k = \frac{1}{n_k}\sum_t^T r_{tk}(\boldsymbol{X}_{t,h} - \bar{\boldsymbol{x}}_k)(\boldsymbol{X}_{t,h} - \bar{\boldsymbol{x}}_k)'$ // [Bishop 2006 Eqs. 10.49,10.51-53]
    $\boldsymbol{\mu}_k = \frac{\beta_0\boldsymbol{m}_0 + \boldsymbol{m}_k\bar{x}_k}{\beta_0 + n_k}$; $\alpha_k = e^{\psi(\alpha_0+n_k)-\psi(\sum_k\alpha_0+n_k)}$ // [Bishop 2006 Eqs. 10.60-62]
    $\boldsymbol{R}_k = \boldsymbol{R}_0 + n_k\boldsymbol{S}_k + \frac{\beta_0 n_k}{\beta_0+n_k}(\bar{\boldsymbol{x}}_k - \boldsymbol{m}_0)(\bar{\boldsymbol{x}}_k - \boldsymbol{m}_0)'$
    $\ln[\rho(\boldsymbol{X}_{t,h}|\boldsymbol{\theta}_{h,Z_h,L_t})] = \ldots$ // [Bishop 2006 Eq. 10.46]
  end
  $N = N + 1$; $Z_{i\neq N} = 0$; $Z_N = 1$
  $\boldsymbol{\theta}_{Z_{h,n}} = \{\boldsymbol{\alpha},\boldsymbol{\mu},\boldsymbol{R}\}$
  StopLearning = FALSE
end

© 2017 Tous droits réservés.
lilliad.univ-lille.fr

Thèse de Matteo Causo, Lille 1, 2017

// VBEM HMM : [Ji et al. 2006 Eqs. 21, 42, 26, 43]
$W_{ij}^{A} = \omega_{ij}^{h-1} + u_{ij}^{A}$; $A_{ij}^{*} = e^{\psi\left(W_{ij}^{A}\right) - \psi\left(\sum_j W_{ij}^{A}\right)}$; $W_i^B = Z_i + u_i^B$; $B_i^{*} = e^{\psi\left(W_i^B\right) - \psi\left(\sum_i W_i^B\right)}$
// Greedy forward pass
$\gamma_{(h|h-1)} = \boldsymbol{A}^{*'}\gamma_{(h-1|h-1)}$; $\gamma_{(h|h)} = \gamma_{(h|h-1)} \odot \boldsymbol{B}^{*} \oslash \mathbf{1}\gamma_{(h|h)}$
$\xi_{(h-1|h)_f} = \left[\gamma_{(h-1|h-1)} \odot \gamma_{(h|h)_f} \oslash \boldsymbol{A}^{*'}\gamma_{(h-1|h-1)}\right]\mathbf{1} \odot \boldsymbol{A}^{*}$; $\gamma_{(h-1|h)_f} = \xi_{(h-1|h)_f}\mathbf{1}'$
$\xi_{(h-1|h)_{f_{ij}}} = \delta\left(\xi_{(h-1|h)_{f_{ij}}} - \max\left(\xi_{(h-1|h)_f}\right)\right)$
$\omega_i^h = \gamma_{(h-1|h)_{f_i}} = \delta\left(\gamma_{(h-1|h)_{f_i}} - \max\left(\gamma_{(h-1|h)_f}\right)\right)$ // Classification result
if *StopLearning == FALSE*
$\omega_{ij}^{h} = \omega_{ij}^{h-1} + \xi_{(h-1|h)_f}$ // VBM HMM
StopLearning = (criteriaStopLearning == TRUE)
end

The VBEM employs the Dirichlet distribution as conjugate prior for the multinomial distribution as described in Equation 2:12. To compute the expectations during the VBE step, the *digamma* $\psi(\cdot)$ function [Abramowitz et al. 1965] is exploited. It is defined as the derivative of the log-Gamma function and is hard to compute analytically. Therefore, it is generally approximated even in software-based implementations. Another contribution of the present PhD thesis work is the optimization of the digamma function computation in order to simplify its HW implementation. Indeed, the next Chapter is devoted at demonstrating by means of a proof-of-concept HW implementation that the proposed on-line VBEM HMM GMM meets the specifications for tackling Big Data elaboration with an Ultra-Low-Power prospective. In this way, intelligence moves close to data sources and we can target portable, wearable and implantable electronics applications.

3.6 Distribution and local elaboration

Another key feature of the proposed solution is its spatial distribution. Several application scenarios for portable, wearable and implantable electronics require intelligence to be as close as possible to data sources. Intuitively, local elaboration of small amounts of data, further simplifies data structures and local computation at the expense of an increased communication overhead [Attiya and Welch 2004].

Our objective is to define a distributed formulation for Algorithm 2 regardless of the network topology. We only assume that there is no centralized arbitration and that each node has a complete view of the system. In fact, thanks to the simplifications discussed above and in particular to the separation between GMM and HMM, the algorithm may be distributed in a flexible manner. Possible scenarios are: (i) several GMM nodes connected to a single HMM node; (ii) several GMM nodes connected to a restricted circle of HMM nodes and (iii) each node performing the whole HMM GMM.

3.6.1 Distribution for the GMM

The GMM distribution follows from the spatial pyramid extension for generative models proposed in [Dixit et al. 2011]. Generally, the spatial pyramid matching is employed in image classification. In particular, an image is repeatedly subdivided in smaller regions for which local features are extracted at increasingly fine resolution and are merged at high level. In our case, each input sequence $\boldsymbol{S}_h^{DxT}$ is divided into M subsequences $\boldsymbol{S}_{h,m}^{D/MxT}$ each of which is assigned to one GMM node. Therefore a collection of M models $p\left(\boldsymbol{S}_{h,m}|\boldsymbol{\theta}_h\right)$ is produced and the overall model log-likelihood is computed by averaging as in Equation 3:10.

$$\log p(\boldsymbol{S}_h|\boldsymbol{\theta}_h) = \frac{\mathbf{1}}{\boldsymbol{M}}\sum_{\boldsymbol{m=1}}^{\boldsymbol{M}} \log p\left(\boldsymbol{S}_{h,m}|\boldsymbol{\theta}_h\right)$$

Equation 3:10 – Log-likelihood for the spatial pyramid.

In [Hua and Li 2016] this is generalized for the distributed variational framework if a centralized fusion center is identified. In our case, each GMM node contribution is merged with the others at the HMM node. This is still a centralized solution that is advantageous, however, if data sources are spatially far apart and, instead of raw data, only the underling GMM parameters are transmitted.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Matteo Causo, Lille 1, 2017

3.6.2 Distribution for the HMM

As for the GMM distribution, the HMM global parameters estimation can be obtained by averaging all local optima calculated at each HMM node. In [Hua and Li 2016], two fully distributed Variational Bayesian algorithms have recently been proposed for decentralized information processing and inference tasks over arbitrary Wireless Sensor Networks (WSNs) having limited communication capabilities. Their objective is to perform almost as well as the centralized solution without relying on a prior knowledge of the network topology, but only on local data and on information obtained form one-hop neighbors. Although this is the ultimate goal for every distributed system and those in [Hua and Li 2016] are extremely interesting solutions, we have relaxed their constraints by means of the above-mentioned assumptions. In fact, our objective is to perform at least as well as the centralized solution because we expect to increase error resilience and to speed up convergence by exploiting redundant connections. Therefore, we envisage a network as in the second scenario, where several GMM nodes are connected to few chosen HMM nodes for which the diffusion-based approach for distributed estimation with unreliable data proposed in [Pereira et al. 2013] and generalized for the Variational framework can be exploited. In particular, for each sequence $\boldsymbol{S}_h$, the greedy forward pass locally computes the sufficient statistics and diffuses them over closer neighbors, which merge them by averaging, apply Equation 3:8 and update their local estimates. The more connected the network, the larger the number of closer neighbors, the wider the system vision and the faster the convergence rate.

3.7 Test and validation

Each simplification of the VBEM HMM GMM proposed in this work has been validated by means of software-based tests aimed at evaluating its performance improvement with fixed final classification accuracy. Below we report three particularly meaningful results we obtained with both synthetic data and real-world applications.

3.7.1 Benchmarking against the state-of-the-art in recovering the hidden model

In this test we compare the execution time required by the proposed approach with respect to the state-of-the-art batch and on-line VBEM HMM GMM in a MATLAB simulation. In particular, the batch algorithm is derived from [Ji et al. 2006] and the on-line one follows from Equation 2:13 according to [Broderick et al. 2013].

# HMM States	$\boldsymbol{N = 3}$
Initial state prob.	$\boldsymbol{\pi} = [.3, .45, .25]$
State transition prob.	$\boldsymbol{A} = \begin{bmatrix} .59 & .17 & .24 \\ .15 & .83 & .02 \\ .25 & .18 & .57 \end{bmatrix}$
# Gaussian components	$\boldsymbol{K} = [2, 1, 2]$
Data dimensionality	$D = 3$
Mixing coefficients	$\boldsymbol{\alpha}_1 = [.04, .96]; \boldsymbol{\alpha}_2 = [1]; \boldsymbol{\alpha}_3 = [.53, .47]$
Gaussian means	$\boldsymbol{\mu}_{11} = \begin{bmatrix} -2 \\ -10 \\ -7 \end{bmatrix}; \boldsymbol{\mu}_{12} = \begin{bmatrix} -5 \\ -4 \\ -9 \end{bmatrix}; \boldsymbol{\mu}_{21} = \begin{bmatrix} -1 \\ -10 \\ -1 \end{bmatrix}; \boldsymbol{\mu}_{31} = \begin{bmatrix} 1 \\ -4 \\ 8 \end{bmatrix}; \boldsymbol{\mu}_{32} = \begin{bmatrix} 4 \\ 0 \\ 8 \end{bmatrix}$

Table 3:1 Toy HMM GMM.

A big amount, 4M, of synthetic data arranged into 16k sequences of 256 data each, is generated by the toy HMM GMM in Table 3:1 and it is presented to the three approaches. They have no clue about the underlined model and their objective is to infer it without any supervision but some reasonable initial priors. The final classification accuracy for each approach is evaluated by comparisons with the toy model. In particular, we do not only require that similar data are classified in the same way, but the number of recognized HMM states and Gaussian components per state has to be the same as for the toy model. The overall test is run on an 8 cores Intel Xeon CPU E5–2643 @ 3.30GHz with 192GB of RAM and the execution time per approach is automatically computed. Results are conveniently summarized in Figure 3:6.

The batch approach elaborates the whole amount of data at once. It converges after 25 iterations, i.e. it is relatively fast because of the ideality of the toy model. In fact, the execution time for the overall variational computations concerns just 1.68% of the total. The rest is spent for passing through the data by means of the FB.

The on-line approach divides input data into sequences and elaborates them as mini-batches. This involves a computational overhead and implies an increase in the execution time for the overall variational computation at 8.10% of the total, as for each sequence the VBEM HMM GMM run until convergence. This could be reduced by employing the on-line approach in [Sato 2001], but also in this case the FB will result critical as it takes most of the execution time. For what concerns the classification accuracy, the on-line

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

approach achieves 97.21% that is slightly less than the 100% obtained by the two other approaches. This could be explained by the fact that on-line algorithms converge generally faster than batch ones but, for fixed sequences size T, their convergence tends to oscillate around their limiting values with variances proportional to T [Khreich et al. 2012].

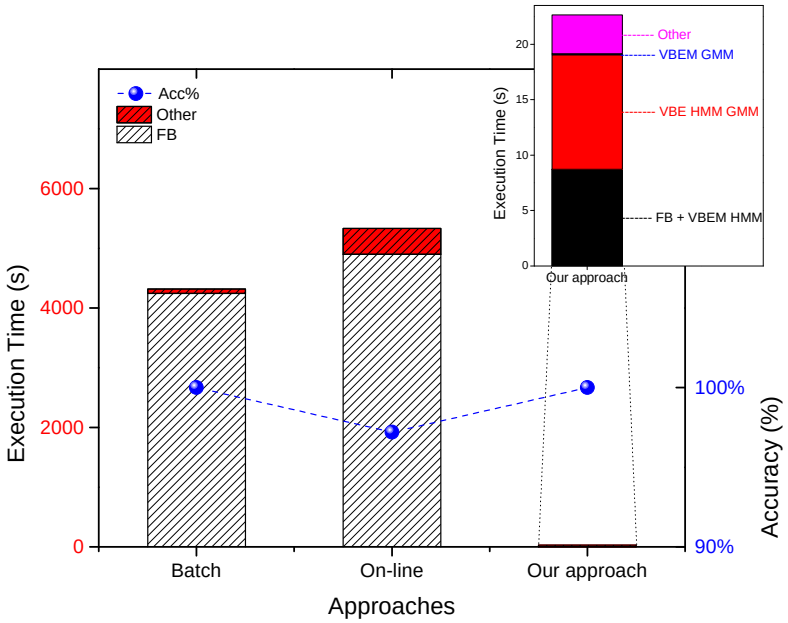


Figure 3:6 Comparisons of results for the synthetic application.

After having applied the aforementioned simplifications, our approach performs learning and classification on the HMM GMM parameters and the VBEM HMM does not see directly the data. Therefore its convergence results incrementally improved, i.e. independent on T as in [Sato 2001]. Finally, the massive simplifications adopted for the FB effectively reduce the overall execution time of two orders of magnitude only with algorithm level considerations and manipulations.

3.7.2 Distribution and Noisy images classification

In this test we compare several distribution configurations of our approach in order to study how distribution improves noise resilience in a real-world application like the noisy images classification. Our approach starts without any knowledge about its task and the data it is going to receive, but some initial priors. Neither the model structure is known a priori, as it is learned in a data-driven and unsupervised manner. For these reasons, we are not aiming to a semantic digit classification for competing with other powerful supervised approaches. Instead we arbitrarily select a subset of instances of just two digits from the MNIST handwritten digit database, Figure 3:7a; randomly arrange them to form a sequence of 100 elements; summarize in Table 3:4 and depict in Figure 3:7c the classification results of our approach arranged according to Table 3:3.

Metrics	Description	Formula
False Positive (FP)	Mistakenly Selected	
True Negative (TN)	Correctly Rejected	
False Negative (FN)	Mistakenly Rejected	
True Positive (TP)	Correctly Selected	
Sensitivity	Correctly Selected / (Correctly Selected + Mistakenly Rejected)	TP / (TP + FN)
Specificity	Correctly Rejected / (Correctly Rejected + Mistakenly Selected)	TN / (TN + FP)
Accuracy	(Correctly Selected + Correctly Rejected) / Total	(TP + TN) / (TP + FP + TN + FN)

Table 3:2 Statistical metrics for evaluating a test.

For evaluation purposes, we adopted conventional metrics based on sensitivity, specificity and accuracy, which are statistical measures of the test performance [Zhu et al. 2010]. For instance, in a binary classification there are two possible classes, i.e. categories of data that share common properties. If we refer to the coin-tossing example in Section 2.1.2, the classifier identifies the coin the current set of tosses belongs to according to the hidden property that is the coin biases. Therefore, there are two classes, A and B and the statistical metrics for the evaluation are described in Table 3:2. If a set of tosses is classified in the class A but it actually belongs to the class B, it results to be mistakenly selected from class A and mistakenly rejected from class B. Conversely, it results

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

correctly selected from the class A while correctly rejected from class B, if it actually belongs to the class A. In particular, sensitivity for the class A indicates how well the test predicts the class A; specificity for the class A measures how well the test predicts the other category, i.e. the class B and accuracy measures how well the test predicts both categories. Hence, a good classification test always results with high values for all the three metrics. For the class A, if sensitivity is high and specificity is low then, class B must be reexamined to eliminate false positives. Conversely, if sensitivity is low and specificity is high then, class A must be reexamined to eliminate false negatives.

Configurations B, D and E require the input digits to be injected with random noise by inversing 1% of the digit's pixels, Figure 3:7b. In particular for configuration E, nodes are assigned with the same digit but different random noise to test their cleaning up ability [Barber 2011].

A	Centralized	Single HMM node and single GMM node
B	Centralized w/ noisy input	
C	GMM distributed	Single HMM node with two GMM nodes
D	GMM distributed w/ noisy input	
E	Fully distributed w/ noisy input	Two HMM nodes with two GMM nodes each

Table 3:3 Distribution configurations.

Config.	Sensitivity	Specificity	Accuracy
A	1.00	1.00	1.00
B	0.96	1.00	0.98
C	1.00	1.00	1.00
D	0.86	0.98	0.88
E	1.00	1.00	1.00

Table 3:4 Tests on Distribution and Noisy images.

In configuration A the single node is able to correctly classify the input images by recognizing the right number of HMM states, i.e. one state per digit. However, when presented with noisy images like in configuration B, it still classifies digits correctly because its accuracy is 98%, while its sensitivity results of 96%. In fact, the digit 1 is spread on several HMM states according to their instances dissimilarity amplified by the noise. Two distinct states are improperly reserved for the digit 1, thus causing False Negatives. Therefore, the learned model structure is wrong, but accuracy stays high as digits are consistently classified.

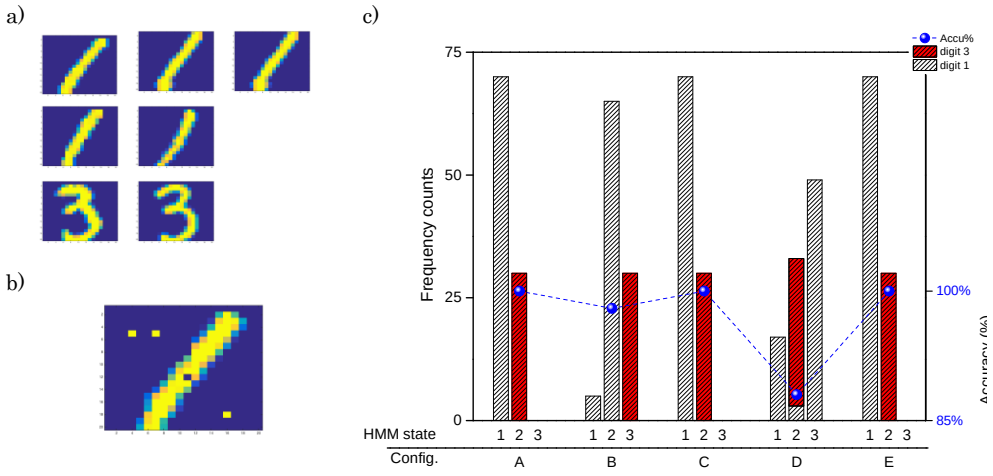


Figure 3:7 Tests on Distribution and Noisy images.

(Set of digits chosen from MNIST handwritten digit database (a). Noisy digit obtained by injection of 1% random noise (b). Comparison of results for the real-world application (c). Letters from A to E correspond to various configurations used to test our approach. In particular, it learns the model structure (HMM states and Gaussian components) and classifies input data accordingly. Classification results are expressed in frequency counts and final accuracy with respect to the true digits classification.)

Similarly, in configuration C our approach performs correctly but in D, the same problem occurs, leading to both a wrong model structure and a poorer accuracy of 88%. Some digits are misclassified and specificity results of 98%. In fact, GMM nodes are responsible for two distinct parts of the same image and if their classification is locally affected by random noise, merging their contributions at the HMM node leads to misclassifications. Instead, configuration E is more robust against noise because HMM nodes

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Matteo Causo, Lille 1, 2017

are provided with the same underlined knowledge, i.e. the digit, even if affected by different random noise. Therefore, merging by averaging at the HMM node is cleaning up the input noise and allows for exact classifications.

Distributed systems perform better as the HMM nodes work independently on the same model learnt from local data. This sounds surprising, but intuitively only the true model is able to satisfactorily explain the data it generates. Therefore, even a small amount of data brings with it the information about the hidden model. Certainly, the probability of under-fitting decreases while the number of data increases. Nevertheless, the Bayesian framework allows information to merge by averaging and to converge to the true solution that maximizes the system welfare.

As preliminary result to be further investigated and generalized, distribution allows for local simple computations but a distribution to all costs and drastically widespread results to be weak against noise. Conversely, an upright redundancy makes a distributed configuration more immune and robust against noise than a centralized one.

3.7.3 EMG-based gesture recognition

In this test we compare our approach with the state-of-the-art in [Benatti et al. 2015] on a real-world application, i.e. the Human-Machine Interface for gesture recognition through Electromyography (EMG) for wearable and portable electronics.

EMG signals are the superposition of the action potentials of the muscle tissue cells occurring during a voluntary contraction. Therefore, EMG is widely employed in gesture-based upper-limb prosthetics control. Recently it is gaining interest in the consumer market for gesture recognition and natural interface applications where surface contact electrodes and appropriate conditioning circuitry can acquire the resulting electrical activity and perform real-time processing. Unfortunately, surface EMG signals are affected by several sources of interference and electrodes positioning compromises EMG repeatability. Therefore, Machine Learning algorithms have been employed in this highly variable scenario. Generally, active sensors and bench-top acquisition systems are used for signal acquisition and the data processing is performed off-line on a PC with classification accuracy over 90%.

In [Benatti et al. 2015] a highly-configurable and scalable system with an on-board real-time signal processing for pattern recognition is provided. The EMG gesture dataset is acquired from four healthy subjects by placing the EMG electrodes on their forearm. During the acquisition sessions, subjects repeated each possible gesture four times; maintained the gesture contractions for 3 seconds and separated each gesture with 3 seconds of hand relaxation, i.e. the rest position. The set of possible gestures consists of power grip; precision grasp; open hand; pointed index and flexion/extension of the wrist. Authors segmented and labeled the EMG collected data by applying a threshold to detect when the gesture starts. They trained a Support Vector Machine (SVM) offline on MATLAB by using 10% of the dataset. The resulting support vectors (SVs) used to discriminate the gestures to be recognized, were stored in the FLASH memory of the MCU and used for real-time classification. The recognition accuracy was computed by applying the resulting SVMs to the continuous stream of acquired gestures opportunely labeled. The average accuracy in the recognition of the seven gestures was 89.2% with a maximum of 92%.

We tested our approach on the same EMG dataset as in [Benatti et al. 2015]. Our approach however performs unsupervised and on-line learning. Indeed, the proposed Algorithm 2 starts processing the EMG data in sequences. It is unaware of its task or the data is going to receive, but it classifies them and learns the underlined model of the reality on-the-fly. In [Benatti et al. 2015], the EMG dataset is split into training and testing parts. Conversely, we provide our classifier with the whole dataset and arrange the learning to stop at the end of the training part, so that testing is fair for the two approaches. We employed the centralized solution in configuration A in Table 3:3 because we are not evaluating the distribution properties of our system.

Results are summarized in Table 3:5 and in particular our approach has a higher classification accuracy than the SVM employed in [Benatti et al. 2015]. Our approach classifies correctly all the possible gestures because it results with a very high value of specificity. Nevertheless, it creates more HMM states than needed to represent the gestures according to the sensitivity metric. A semi-supervision of the model structure will improve both the sensitivity value and overall classification accuracy of our approach.

	[Benatti et al. 2015]	**Our approach**
Application	Gesture recognition from EMG 4ch	
Type	MATLAB / Embedded System	MATLAB
Focus	Real-time Human Machine Interface	Distributed Online General-Purpose Learning-on-a-Chip
Features	-	-
Learning algorithm	Linear SVM	VBEM GMM HMM
Learning / Testing	Supervised Offline & Ex-situ / Online & In-situ	Un/Semi-supervised Online & In-situ / Online & In-situ

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Accuracy (%)	92 (max)	98
Sensitivity (%)	-	88
Specificity (%)	-	99

Table 3:5 Test on EMG gesture recognition.

3.7.4 Algorithm complexity versus model complexity

In this test we aim to study the impact of complex models on the accuracy and the complexity in terms of execution time of our approach. Therefore, we tested the ability of a single node, configuration A, to correctly classify ten different digits chosen from the MNIST handwritten digits database and randomly arranged into a sequence of 100 elements. Classification results are summarized in Table 3:6. The single node is performing generally well with an accuracy of 98%, but it misclassifies some digits, i.e. specificity at 97%. In a distributed arrangement like configuration C, our approach successfully performs learning and classification as already remarked in the previous application. Intuitively, the complexity of our approach is somehow related to both the model complexity in terms of the number of HMM states N and the system complexity in terms of the number of nodes M. Generally, the higher is the model complexity, the poorer is the accuracy. Conversely, an optimal distribution scheme increases the collaboration between nodes so that not only the complexity of the system increases, but also its accuracy.

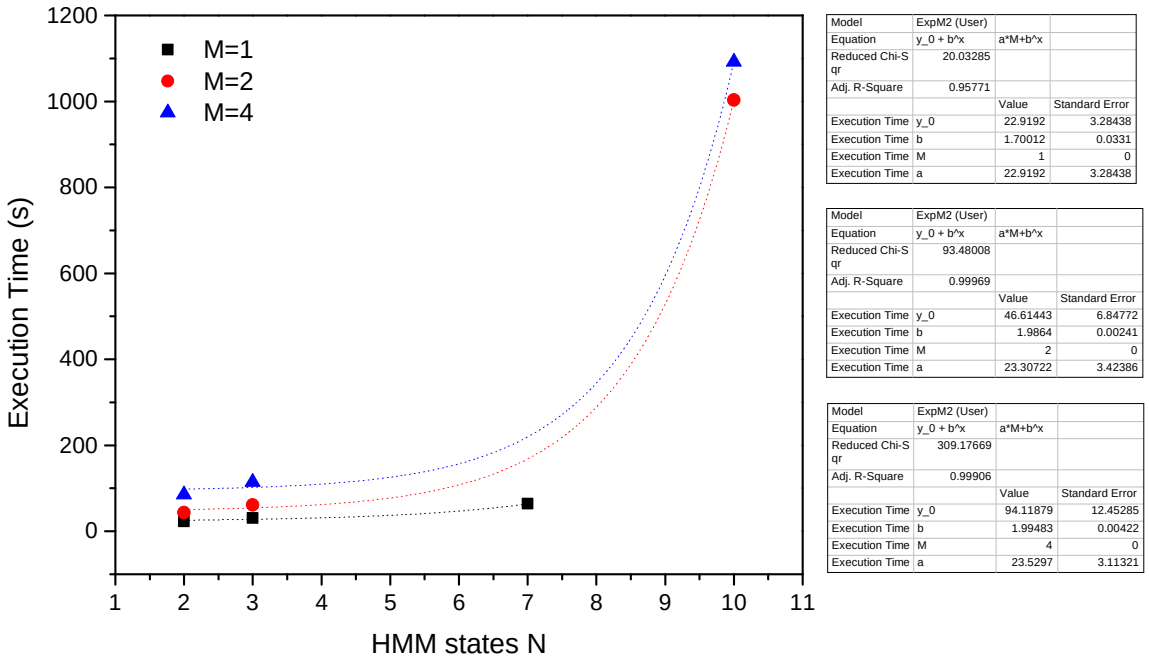
Config.	Sensitivity	Specificity	Accuracy
A	1.00	0.97	0.98
C	1.00	1.00	1.00

Table 3:6 Test on 10 MNIST handwritten digits (0 to 9).

To quantify this finding we compared the complexity of our approach in terms of execution time for the applications in the present Section, namely the synthetic application in Section 3.7.1; the Noisy Images in Section 3.7.2; the EMG-based gesture recognition in Section 3.7.3 and the ten MNIST handwritten digits. We studied the relation between the execution time required for classifying 16k images and the number of HMM states N or the number of nodes M.

$$c = 23 \cdot M + 2^N$$

Equation 3:11 – Resulting formula for the complexity estimation according to N and M.



Model	ExpM2 (User)		
Equation	y_0 + b^x	a*M+b^x	
Reduced Chi-Sqr	20.03285		
Adj. R-Square	0.95771		
		Value	Standard Error
Execution Time	y_0	22.9192	3.28438
Execution Time	b	1.70012	0.0331
Execution Time	M	1	0
Execution Time	a	22.9192	3.28438

Model	ExpM2 (User)		
Equation	y_0 + b^x	a*M+b^x	
Reduced Chi-Sqr	93.48008		
Adj. R-Square	0.99969		
		Value	Standard Error
Execution Time	y_0	46.61443	6.84772
Execution Time	b	1.9864	0.00241
Execution Time	M	2	0
Execution Time	a	23.30722	3.42386

Model	ExpM2 (User)		
Equation	y_0 + b^x	a*M+b^x	
Reduced Chi-Sqr	309.17669		
Adj. R-Square	0.99906		
		Value	Standard Error
Execution Time	y_0	94.11879	12.45285
Execution Time	b	1.99483	0.00422
Execution Time	M	4	0
Execution Time	a	23.5297	3.11321

Figure 3:8 Algorithm complexity evaluations.

(Complexity analysis for the synthetic application; the Noisy images; the EMG gesture recognition and the ten MNIST handwritten digits applications in various nodes configurations. On the right-side the fitting parameters for the three curves, which show the common behavior of Equation 3:11.)

© 2017 Tous droits réservés.

Thèse de Matteo Causo, Lille 1, 2017

By fitting the curves in Figure 3:8, we derived the interesting relation in Equation 3:11 where c is the complexity calculated in terms of execution time in seconds, which is linearly dependent on M and exponentially on N. Therefore, optimizations should envisage distributed solutions for locally computing small amounts of data and for resulting with a reduced model complexity, i.e. N has to be upperly bounded.

3.8 Conclusions

The state-of-the-art VBEM HMM GMM has two principal problems. Firstly, data structures encode redundant information that implies a complexity overhead. Secondly, some parameters depend linearly on the number of sequences H. In a streaming application, H goes to infinity and makes unfeasible to map its architecture into a finite physical device.

We proposed a less complex solution as it strongly benefits from the on-line non-parametric computation: N is incremented progressively from 1; data structures are simplified; VBE HMM, FB and VBM HMM do not need to see the data; VBE HMM GMM, VBE HMM, FB and VBM HMM require one iteration per sequence and the VBEM GMM runs until convergence only when a new HMM state is detected. Furthermore, we improved the FB block by definitely eliminating its inherent dependence on H and by introducing the greedy forward pass that takes advantage of the on-line elaboration. In this way, we meet this PhD thesis objective of simplifying and enhancing the on-line VBEM HMM GMM. In fact, we gain two orders of magnitude in complexity reduction with only algorithm level optimizations, as shown in Figure 3:6.

Finally we defined and studied the spatial distribution for the proposed on-line VBEM HMM GMM and we drew important considerations on its noise cleaning up properties. We derived the paramount relation in Equation 3:11 to trade-off distribution and complexity by achieving a valid final classification accuracy.

We have tested the proposed Algorithm 2 with both synthetic and real-world applications, such as the MNIST handwritten digits recognition and EMG-based gesture recognition. We generally showed better results with respect to the state-of-the-art solutions.

In the next Chapter, we demonstrate that the proposed algorithm can finally benefit of a further order of magnitude in complexity reduction through traditional HW optimization techniques. Indeed, we made its HW implementation both feasible and attractive and therefore suitable for wearable, portable and implantable electronics. In particular, we propose a proof-of-concept on a FPGA platform and implement the on-line VBEM HMM GMM by means of conventional digital CMOS technology.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Matteo Causo, Lille 1, 2017

References

Milton Abramowitz, Irene a. Stegun, and David Miller. 1965. Handbook of Mathematical Functions With Formulas, Graphs and Mathematical Tables (National Bureau of Standards Applied Mathematics Series No. 55). *J. Appl. Mech.* 32, 1 (1965), 239. DOI:http://dx.doi.org/10.1115/1.3625776

Hagit Attiya and Jennifer Welch. 2004. Distributed Computing. (2004), 414. DOI:http://dx.doi.org/10.1002/0471478210

David Barber. 2011. Bayesian Reasoning and Machine Learning. *Mach. Learn.* (2011), 646. DOI:http://dx.doi.org/10.1017/CBO9780511804779

Simone Benatti et al. 2015. A Versatile Embedded Platform for EMG Acquisition and Gesture Recognition. *IEEE Trans. Biomed. Circuits Syst.* 9, 5 (October 2015), 620–630. DOI:http://dx.doi.org/10.1109/TBCAS.2015.2476555

Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning*,

Tamara Broderick, Nicholas Boyd, Andre Wibisono, Ashia C. Wilson, and Michael I. Jordan. 2013. Streaming Variational Bayes. In *NIPS*. 1–9.

V.V. Digalakis. 1999. Online adaptation of hidden Markov models using incremental estimation algorithms. *IEEE Trans. Speech Audio Process.* 7, 3 (1999), 253–261. DOI:http://dx.doi.org/10.1109/89.759031

Mandar Dixit, Nikhil Rasiwasia, and Nuno Vasconcelos. 2011. Adapted Gaussian Models for Image Classification. *Comput. Vis. Pattern Recognit.* (2011).

German Florez-larrahondo, Susan Bridges, and Eric a Hansen. 2001. Incremental Estimation of Discrete Hidden Markov Models Based on a New Backward Procedure. *Engineering* (2001), 758–763.

J.a Grice, R. Hughey, and D. Speck. 1997. Reduced space sequence alignment. *Comput. Appl. Biosci.* 13, 1 (1997), 45–53.

J.A. Hartigan and M.A. Wong. 1979. Algorithm AS 136: A K-Means Clustering Algorithm. *J. R. Stat. Soc. C* 28, 1 (1979), 100–108. DOI:http://dx.doi.org/10.2307/2346830

Junhao Hua and Chunguang Li. 2016. Distributed Variational Bayesian Algorithms Over Sensor Networks. *IEEE Trans. Signal Process.* 64, 3 (2016), 783–798. DOI:http://dx.doi.org/10.1109/TSP.2015.2493979

Shihao Ji, Balaji Krishnapuram, and Lawrence Carin. 2006. Variational Bayes for continuous hidden Markov models and its application to active learning. *IEEE Trans. Pattern Anal. Mach. Intell.* 28, 4 (2006), 522–532. DOI:http://dx.doi.org/10.1109/TPAMI.2006.85

Wael Khreich, Eric Granger, Ali Miri, and Robert Sabourin. 2012. A survey of techniques for incremental learning of HMM parameters. *Inf. Sci. (Ny).* 197 (2012), 105–130. DOI:http://dx.doi.org/10.1016/j.ins.2012.02.017

Wael Khreich, Eric Granger, Ali Miri, and Robert Sabourin. 2010. On the memory complexity of the forward-backward algorithm. *Pattern Recognit. Lett.* 31, 2 (2010), 91–99. DOI:http://dx.doi.org/10.1016/j.patrec.2009.09.023

Dahua Lin. 2013. Online Learning of Nonparametric Mixture Models via Sequential Variational Approximation. *Adv. Nueral Inf. Process. Syst. 26 (Proceedings NIPS)* (2013), 1–9.

JB Moore and J.J. Ford. 1998. Reduced complexity on-line estimation of Hidden Markov Model parameters. *Proc. 1998 Int. Conf.* (1998).

Gene Ott. 1967. Compact Encoding of Stationary Markov Sources. *IEEE Trans. Inf. Theory* 13, 1 (1967), 82–86. DOI:http://dx.doi.org/10.1109/TIT.1967.1053960

S.S. Pereira, R. Lopez-Valcarce, and A. Pages-Zamora. 2013. A Diffusion-Based EM Algorithm for Distributed Estimation in Unreliable Sensor Networks. *IEEE Signal Process. Lett.* 20, 6 (2013), 595–598. DOI:http://dx.doi.org/10.1109/LSP.2013.2260329

Masa-aki Sato. 2001. Online Model Selection Based on the Variational Bayes. *Neural Comput.* 13, 7 (2001), 1649–1681. DOI:http://dx.doi.org/10.1162/089976601750265045

Srinivas Sivaprakasam and K. Sam Shanmugan. 1995. ecursion based HMM for rst Errors in Digital Channels. *Electr. Eng.* (1995), 1054–1058. DOI:http://dx.doi.org/10.1109/GLOCOM.1995.502566

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Matteo Causo, Lille 1, 2017

B. Stenger, V. Ramesh, N. Paragios, F. Coetzee, and J.M. Buhmann. 2001. Topology free hidden Markov models: application to background modeling. *Proc. Eighth IEEE Int. Conf. Comput. Vision. ICCV 2001* 1 (2001), 294–301. DOI:http://dx.doi.org/10.1109/ICCV.2001.937532

Wen Zhu, Nancy Zeng, and Ning Wang. 2010. *Sensitivity, specificity, accuracy, associated confidence interval and ROC analysis with practical SAS® implementations.*, Baltimore, Maryland.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Chapter 4 HW implementation and FPGA demos

Two orders of magnitude in complexity reduction have been obtained in the previous Chapter through algorithm level considerations and manipulations. In this Chapter, we study the state-of-the-art of Neuro-Inspired HW solutions and propose a proof-of-concept HW implementation for the proposed on-line VBEM HMM GMM. We show a further order in complexity reduction by traditional optimization techniques and innovative contributions. Moreover, we demonstrate the Bayesian Machine Learning is a convenient framework for Ultra-Low-Power (ULP) Neuro-Inspired computing platforms.

4.1 State-of-the-art HW implementation

The objective of the present PhD thesis work is to approach Neuro-Inspired filed with an Ultra-Low-Power (ULP) prospective in order to provide strictly constrained electronics with learning capabilities. This falls under Learning-on-a-Chip solutions that envisage intelligence all-over the processing even close to data sources. Nevertheless, traditional Machine Learning (ML) relies on state-of-the-art software-based CPUs/GPUs implementations. Therefore an acceleration factor of 10^2 -10^4 is critically needed to achieve orders of magnitude improvement in performance, energy efficiency and compactness [HALO 2015]. Two are the mainstream trends: (i) technology optimizations and (ii) algorithms replacement with next-generation ML.

(i) Technology optimizations assume the neuromorphic engineering approach to be well suited for post-Moore technologies. On-chip memory integration solves the off-chip memory wall through in-memory computing, i.e. interleaving logic and emerging memory nanodevices together like in the human brain's synapses. However, these promising solutions are matter of ongoing research and are not yet mature for mass production because of the heterogeneous processes.

(ii) Many efforts have focused on easier-scalable and less-demanding ML algorithms that can be mapped in conventional CMOS technology. Therefore, they have a smaller time to market (TTM) than exotic solutions that would require a radical revolution of the semiconductor industry.

In the present PhD thesis work, we have deliberately chosen the second option by supporting and adopting the Bayesian ML (BML). In fact, thanks to its probabilistic nature, BML deals with uncertainty by performing learning during inference [Barber 2011]. This represents a huge advantage in facing nanometer processes variation, noise, soft errors, and other non-idealities that are nullifying the intrinsic benefits of massive scaling [Shanbhag et al. 2008]. In this Chapter, we aim at demonstrating, BML results less challenging to accelerate through standard acceleration techniques than Deep Neural Networks (DNNs) because it does not require dense interconnections and opens to other learning schemes than supervised learning. In fact, naïvely mapping traditional ML algorithms to physical hardware fails due to the communications bottleneck: parameters require massive off-chip memories and dedicated hardware solutions become unfeasible or no more advantageous due to the complexity of the training.

We have already demonstrated two orders of magnitude in complexity reduction with only algorithm level optimizations on the state-of-the-art VBEM HMM GMM. We target a further order of magnitude in complexity reduction by traditional optimization techniques. Therefore, we define a proof-of-concept implementation on a FPGA platform in order to result with a practical demo for real-word streaming analytics.

State-of-the-art Learning-on-a-Chip solutions are summarized and compared in Table 4:1. Comparison is however unfair because every approach targets a different application, a different implementation and even a different system architecture with respect to the others. In fact, Learning-on-a-Chip is a recent field and aspires to a consistent HW and algorithm co-optimization to speed up performance or reduce power consumption. If no ML algorithm is a dominant design yet and each ML algorithm is better suited for a specific application, then it results hard to define a baseline for fair comparison of HW implementations. Anyway, our approach is generally valid so that we can target generic streaming analytics applications. In this way, we are able to compare our classification results and our HW implementation performance with every other approach in the state-of-the-art and highlight the strengths of our proposed solution.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Matteo Causo, Lille 1, 2017

Solutions in Table 4:1 can be grouped in accord with their common characteristics. For example, most implementations are on FPGA platform, while only two solutions propose SoC on dedicated ASICs with however different technologies. [Benatti et al. 2015] proposes a real-time Human-Machine Interface (HMI) for Gesture recognition on Electromyography (EMG) recordings, we used for comparison to our algorithm in Section 3.7.3. Although it is an interesting solution, it remains supervised and learning is performed off-line on a PC and only the testing is in real-time. This is common to [Park et al. 2015] and [Zhang and Parhi 2016]. On the contrary, [Biradar et al. 2015] proposes on-line learning and the others unsupervised on-line learning, very similar to our solution. Regarding the evaluation of the classification performance, the accuracy is the most widely used metric. For area or power consumption, some work provides measured or estimated consumption, while others only the FPGA synthesis reports.

		[Park et al. 2016]	[Benatti et al. 2015]	[Zhang and Parhi 2016]	[Biradar et al. 2015]	[Zyarah and Kudithipudi 2016]	[Sasagawa and Mori 2016]	[Ren et al. 2014]	[Jewajinda and Chongstitvatana 2012a]
Application		Gesture recognition from 32x32 images	Gesture recognition from EMG	EEG patient-specific seizure prediction	On-line LTI system identification	MNIST handwritten digit classification	Object tracking	On-line LTI system identification	real-time classification of ECG signals
Type		SoC (65nm)	Embedded System	Algorithm	FPGA	FPGA	SoC (28 nm)	FPGA	FPGA
Focus		Circuital optimization and implementation optimization	Real-time Human Machine Interface	Feature Extraction selection and reduction	Design basic learning modules for an IP catalogue	Memory-centric Spatial Pooler for Hierarchical Temporal Memory (HTM)	High-level video analytics PC sub-system as PCIe add-in card	Low-cost HW implementation 128-way parallel HW	Adaptive feature selection and configurable hardware
Features		Convolutional Layer	-	Power Spectral Density (PSD)	-	MATLAB pre-processing	Luminance integral	-	QRS complex estimation
Learning algorithm		Convolutional Deep Belief Network (CDBN)	Linear SVM	Linear SVM	Back-Propagation on Feed-Forward NN	Cortical learning Algorithm (CLA)	Random Fern classifiers	Mercer Kernel Last Mean Square (KLMS)	cellular compact genetic algorithm (c-cGA) applied to block-based neural network (BbNN)
Learning / Testing		Supervised Offline & In-situ / Online & In-situ	Supervised Offline & Ex-situ / Online & In-situ	Supervised Offline & Ex-situ / Online & Not-Spec.	Supervised On-line & In-situ / On-line & In-situ	Unsupervised On-line & In-situ / On-line & In-situ	Unsupervised On-line & In-situ / On-line & In-situ	Unsupervised On-line & In-situ / On-line & In-situ	Unsupervised On-line & In-situ / On-line & In-situ
Accuracy (%)		95.9 (Best-in-class)	92 (max)	98.68	Qualitative	91	-	-	98.7
SRAM		216 kB	192 kB	-	-	-	-	-	-
Database		640 B	9.6 kB	9.6 kB					
Power consumption		213.1 mW @ 200MHz	86 mW @ 168MHz	-	-	-	15W	-	-
Energy per point classification		-	1.5 uJ (1kHz) 1.08uJ (25Hz)	-	-	-	-	-	-
Logic utilization:	**LUT**	-	-	-	48086	20997	-	123264	9849
	Registers	-	-	-	25450		-	79744	8392
	BlockRAM	-	-	-	10	52	-	384	-
	DSP48E	-	-	-	16		-	1024	14

Table 4:1 State-of-the-art HW implementations.

4.2 Proposed SoC architecture

Simplifications leading to the proposed on-line algorithm make its HW acceleration both feasible and efficient in view of the HW and algorithmic co-optimization. Results shown in Figure 3:6 indicate that the objective of reducing the complexity of the system by two orders of magnitude has been achieved with convenient algorithm level considerations and manipulations. A further improvement is therefore obtained through conventional HW acceleration techniques that lead to the definition, in Figure 4:1, of a small ULP SoC architecture with Learning-on-a-Chip capabilities.

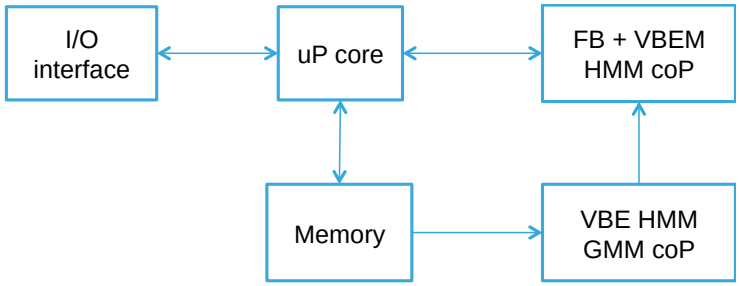


Figure 4:1 Proposed System-on-a-Chip architecture

The inset graph at the upper-right corner of Figure 3:6 gives us an idea about the complexity of the proposed approach building blocks reported in Figure 4:1. In particular, both the Variational Bayesian Expectation Hidden Markov Model Gaussian Mixture Model (VBE HMM GMM) and the Forward Backward + Variational Bayesian Expectation-Maximization Hidden Markov Model (FB + VBEM HMM) are repeatedly executed at each new sequence presentation and they take most of the total execution time. Therefore, dedicated implementations like dedicated CoProcessors (CoPs), can be designed to speed up their execution or to lower their complexity. The Variational Bayesian Expectation-Maximization Gaussian Mixture Model (VBEM GMM) instead, is sporadically run, i.e. only when a new HMM state is detected. Furthermore, a dedicated HW for the VBM GMM is worthless because it requires several complex matrix operations that cannot be simplified with specific assumptions on input data. In fact, we want our approach to be general and so the VBEM GMM is opportunely executed by an ULP uProcessor (uP). Following this philosophy, also the Feature

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Extraction Engine (FEE) that is specific for each application, can be easily reprogrammed if software-based. The Memory block is mainly used as buffer for each sequence of data to feed either the VBE HMM GMM CoP or the uP running the VBEM GMM. In a distributed configuration, like those examined in Table 3:3, the uP retrieves other nodes' contributions to be processed by the the FB + VBEM HMM CoP. This latter incrementally estimates the HMM parameters and performs classification. Finally, the uP manages the output of results, i.e. visualization and transmission.

4.3 FB + VBM HMM CoP design and optimization

After a careful feasibility analysis, we leave the VBE HMM GMM block in SW for the proof-of-concept through the FPGA demos that are the outcome of this Chapter. In fact, this block is not conveniently optimized in HW as it requires to perform several matrix operations, whose simplification and approximation are mathematically impossible without significant penalty in the results and whose optimization follows the developments in the state-of-art solutions like in [Mahale et al. 2016]. Then, this block will be implemented once the whole structure of the SoC identified and validated for the production of an ASIC in 28 nm CMOS technology.

On the contrary, the FB + VBEM HMM results to be conveniently optimized and its implementation represents an innovation compared to the state-of-the-art. In fact, we speed it up for the analysis of continuous streams and optimize its energy consumption. In Figure 4:2, a schematic representation of the FB + VBEM HMM after the simplifications of Chapter 3, is proposed. During the VBM-step, the sufficient statistic ω_{ij}^{h-1} previously computed by means of the Greedy Forward Pass and the indicator $\boldsymbol{Z}$ representing the underlined state in the current sequence $\boldsymbol{S}_h$ of input data, are respectively summed up with the priors $\boldsymbol{u}^A$ and $\boldsymbol{u}^B$. During the VBE-step, the expectations on the resulting $\boldsymbol{W}^A$ and $\boldsymbol{W}^B$ are computed by means of the digamma function, introduced in Section 3.5. In particular, the digamma is applied to elementwise to both structures and it projects their values in the Log-domain. Therefore, the subtractions in the equations of Algorithm 2, based on [Ji et al. 2006 Eqs. 42, 43] read as normalization operations with respect to all the values in the same data structure. Finally, the resulting $\boldsymbol{A}^*$ and $\boldsymbol{B}^*$ are provided to the Greedy Forward Pass block after a Log-to-Normal domain conversion. The Greedy Forward Pass block performs the Forward Filtering Backward Smoothing (FFBS) algorithm and computes the sufficient statistics and the classification results to be employed for the next iteration on the successive input sequence $\boldsymbol{S}_{h+1}$.

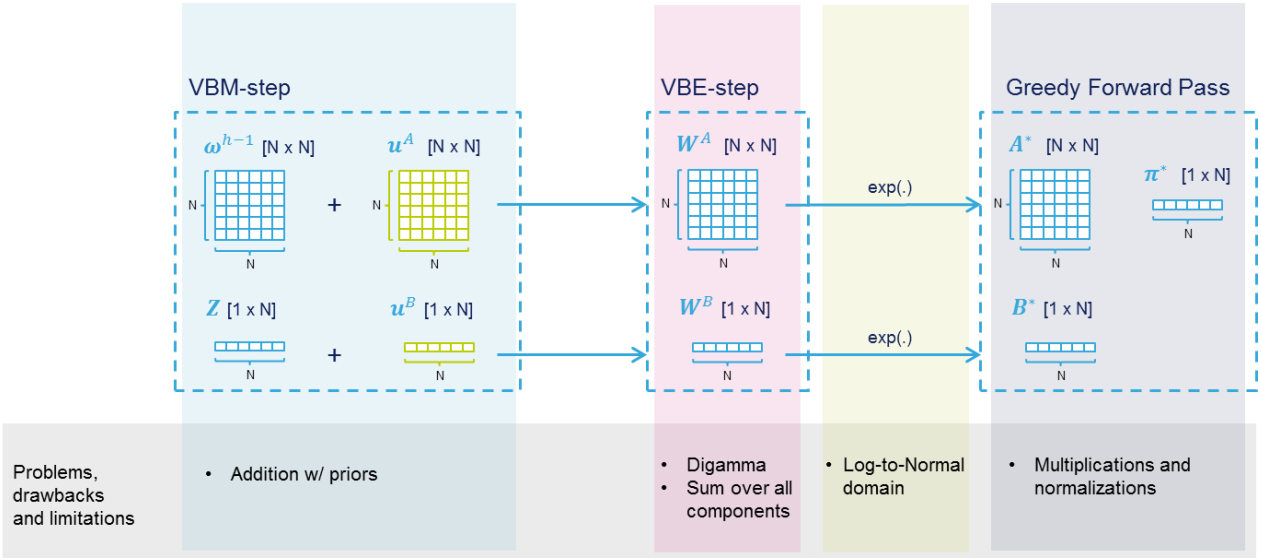


Figure 4:2 Schematic representation of the FB + VBEM HMM procedures.

We optimized the FB + VBEM block in Figure 4:2 though important contributions. We conveniently chose the prior $\boldsymbol{u}^A$ to equal the all-one matrix and $\boldsymbol{u}^B$ to disappear so that, according to the formulation in Equation 3:8, $\boldsymbol{W}^A$ and $\boldsymbol{W}^B$ structures result as frequency counts. In this way, the computation of $\boldsymbol{A}^*$ simplifies because normalization can be computed independently by rows and the sum over all components of $\boldsymbol{W}^A$ is not needed anymore. Moreover, we modified the Greedy Forward Pass block to perform Log-operations as detailed in Section 4.3.4. Benefits are twofold: multiplications and normalizations become respectively addition and subtraction, and there are no more overheads for the Log-to-Normal domain conversion. Therefore, the VBEM HMM extremely simplifies as shown in Figure 4:3, where for recursive operations, the value in the boxes is by convention the operation result.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

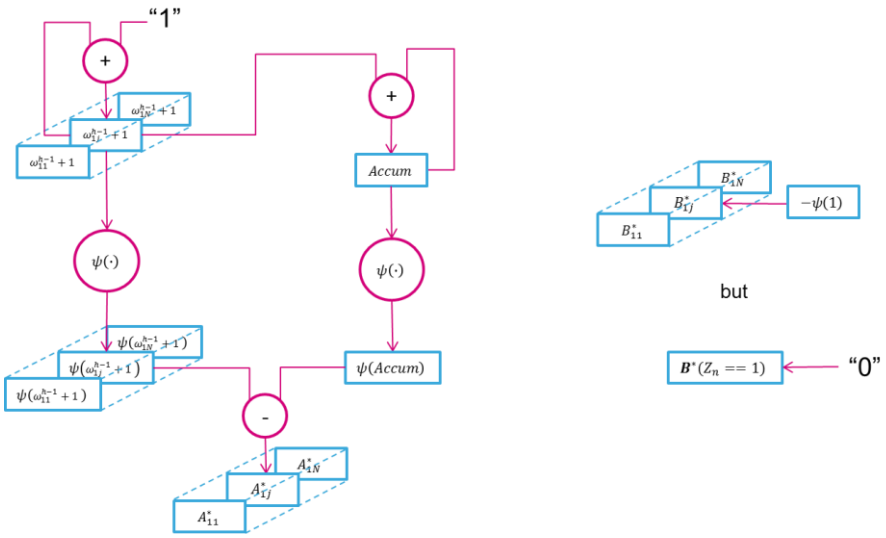


Figure 4:3 Flowcharts for the optimized simplified VBEM HMM.

In particular, the $\boldsymbol{\omega}^{h-1}$ matrix resulting from the previous iteration of the Gready Forward Pass is eleaborated by rows $\boldsymbol{\omega}_i^{h-1}$ and each element ω_{ij}^{h-1} is incremented by one as to sum with the all-one prior $\boldsymbol{u}^A$. At the same time, each row is summed up into an $Accum$ varaible to which the digamma function is applied. The digamma function $\psi(\cdot)$ is also applied elementwise to each element $(\omega_{ij}^{h-1}+1)$ in order to results with $\psi(\omega_{ij}^{h-1}+1)$ that is in the Log-domain. Therefore, the $\boldsymbol{A}^*$ matrix is computed from the difference of each $\psi(\omega_{ij}^{h-1}+1)$ and the digamma of the overall row $\psi(Accum_i)$, i.e. the normalization operation in the Log-domain. For what concerns the computation of the $\boldsymbol{B}^*$ array, our simplifications lead to initialize the $\boldsymbol{B}^*$ array at a constant value $-\psi(1)$, but for the element at the position indicating the recognized HMM state, i.e. the position for which $Z_n == 1$, which takes zero as value.

The VBEM HMM results therefore extremely simplified, but the digamma function may represent a bottleneck for this implementation. Its optimization identifies an innovative contribution of the present PhD thesis work.

4.3.1 Incremental/Decremental Digamma function

In Section 3.5, we introduced the digamma function $\psi(\cdot)$ that is hard to compute analytically. Therefore, it is generally approximated even in software-based implementations[12]. Nevertheless, the digamma function presents the particularly interesting recursive properties in Equation 4:1, which is generally valid and widely exploited for its approximation.

$$\psi(x+1) = \psi(x) + 1/x$$

Equation 4:1 – Digamma incremental recursion formula.

Our observation is that thanks to our choice for the prior $\boldsymbol{u}^A$, we can compute the digamma function incrementally. In other words, we can exploit again the on-line formulation and for each iteration compute the digamma of the incremented ω_{ij}^{h-1} based on the digamma we already computed during the previous iteration. This is represented in Figure 4:4, where each element x_{ij} is incremented and reversed. Finally, it is added to $\psi(x+1)$ in order to obtain $\psi(x+2)$ according to Equation 4:1.

The advantage of employing Equation 3:8 and to work with frequency counts is therefore evident. We provide our machine with the initial value of $\psi(1)$ and successively update it by tacking advantage of the on-line formulation.

However, this mechanism for incrementally computing the digamma function has a key disadvantage. The next value to compute depends on the current one, therefore we lose the cognition on which value we are applying the digamma function to. In the unlikely case the VBEM HMM needs to prune or merge some HMM states, the argument of the digamma function will be impacted. Therefore, we may need to compute its digamma value by starting iteratively from $\psi(1)$. This will take a long time for big arguments and after all, we do not know the exact value of the argument to stop iterating.

[12] Boost C++ Libraries.

© 2017 Tous droits réservés.

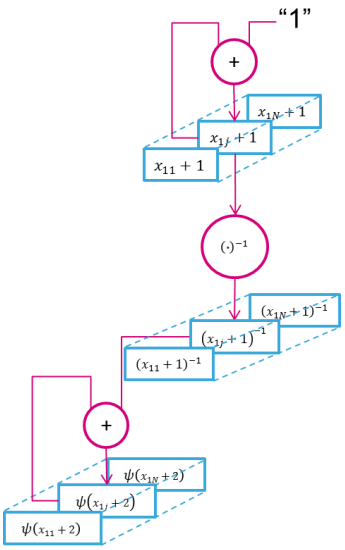


Figure 4:4 Flowchart for the digamma function.

We introduce the possibility of incrementally computing the digamma value for previous values of the argument according to Equation 4:2, i.e. a decremental digamma formulation. In this way, we can face to a prune or a merge of HMM states, still without having cognition of the argument value, rather of the differential steps we need to perform backward.

$$\psi(x) = \psi(x+1) - 1/x$$

Equation 4:2 – Digamma decremental recursion formula.

Equation 4:2 requires the addition in Figure 4:4 to be replaced by subtractions. Therefore, we implement both incremental and decremental digamma on the same HW with a very light overhead. Nevertheless, the complexity in computing the digamma function shifts towards the computation of the reciprocal value.

4.3.2 Incremental/Decremental reciprocal

Our implementation of the digamma function is interesting but still requires a division/reciprocal operation that can be demanding in both time and area/power resources for an HW implementation.

An fast and efficient way of implementing a division is by exploiting the Newton-Raphson method [Möller and Granlund 2011]. In particular, it allows to compute the roots for any given function f defined over the real numbers x; differentiable with derivative f' and with an initial guess x_0 for its root. A better approximation x_1 for the root value is given by $x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$, which results in the recursion formula of Equation 4:3 to be repeated until a sufficiently accurate result is reached.

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Equation 4:3 – Newton-Raphson recursion formula.

Therefore, the reciprocal of a number a is automatically computed if the function f is chosen to be $f = a - 1/x$. The recursion formula easily derives from Equation 4:3 and in particular results $x_{n+1} = x_n(2 - ax_n)$. It consists of only two multiplications and one subtraction, but it has several disadvantages. Firstly, it requires successive iterations in order to improve the approximation error and secondly both convergence and convergence rate depend on the choice of the initial guess for the result.

At first, we try to optimize this method by reducing the uncertainty in the choice of the initial root value. Then, we build on an ad-hoc method that exploits the on-line elaboration and that represents a further contribution of this PhD thesis work. In fact, we overcome all the disadvantages of the Newton-Raphson method because the reciprocal is computed directly and incrementally from the previous value in one iteration and there is no need for guessing a probable initial value for the result. Moreover, our method

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

requires only one multiplication and a subtraction, resulting less exigent in terms of time and area/power resources than the Newton-Raphson method for this specific application.

In particular, our observation is that $a = 1/x_a$ and generally we need to compute $a + 1 = 1/x_{a+1}$. Then we substitute a from the former equation into the latter and we obtain $1/x_a + 1 = 1/x_{a+1}$, which trivially results in Equation 4:4.

$$x_{a+1} = \frac{x_a}{x_a + 1} = \begin{cases} 1 - a + a^2 - a^3 + R & x_a \to \infty \\ 1/2 & \text{if} \quad x_a \to 1 \\ x_a - x_a^2 + x_a^3 + R & x_a \to 0 \end{cases}$$

Equation 4:4 – Reciprocal incremental recursion formula.

R is the error we commit by approximating the reciprocal x_{a+1} for $a + 1$ with the current reciprocal x_a computed for a. Because a grows incrementally, then x_a tends to 0. Thus, our formulation for the incremental reciprocal is $x_{a+1} = x_a - x_a^2 + R$, where R is $O(x_a^3)$ and the implementation in Figure 4:5 follows.

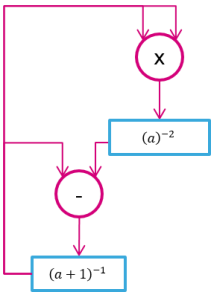


Figure 4:5 Flowchart for computing the reciprocal.

The reciprocal of $a + 1$ is therefore approximately computed with the difference between the reciprocal of a and its square. To esteem the quality of our results with respect to the correct values, we compare both with an infinite precision simulation on MATLAB which result is plotted in Figure 4:6Left. In particular, R results evident and for small values of a, our method's results are not valid. However, if we map the correct results for the reciprocal computation of small values of a, e.g. $a \in [1,16)$, R becomes very small to be considered, as reported in Figure 4:6Right and our approach to the reciprocal computation results valid.

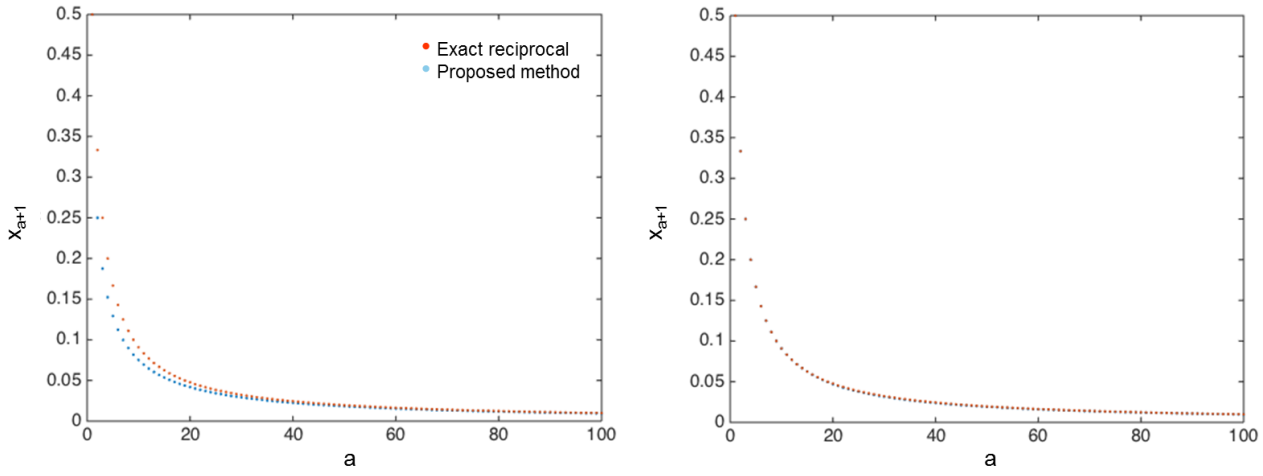


Figure 4:6 MATLAB infinite resources simulation to validate the proposed method.

(The red curve represents exact values, while the blue one the approximated results of the incremental reciprocal, i.e. the proposed method. We compare the accuracy of the iterative approximation (left) vs. iterative approximation + error correction (right).)

In order to implement the decremental digamma, we need to compute the decremental reciprocal, thus we follow the same reasoning as above and we result with the recursive formula $x_{a-1} = x_a + x_a^2 + R$ if $x_a \to 0$. The same flowchart as in Figure 4:5 is exploited, but the subtraction is substituted with an addition. In this case it is however a bit trickier to correct our method results with a LUT for small values of a. In fact, the error the decremental reciprocal commits is bigger than that of the incremental one.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

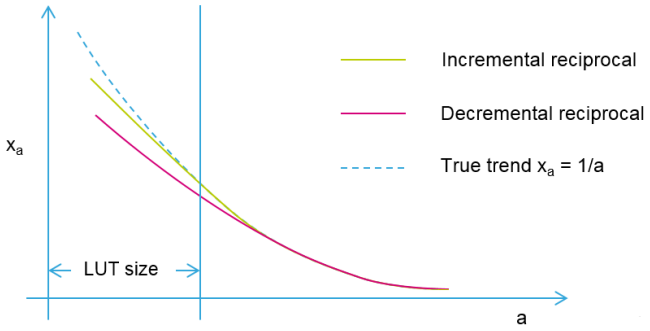


Figure 4:7 Cartoony representation for qualifying the error of our method results.

Therefore, without cognition which a we are computing the reciprocal on, it results difficult to hook the right LUT entry by proceeding backward on a, as shown in Figure 4:7. In fact, we observe that the maximum error we can commit is always at the interface with the LUT following the backward direction, i.e. the decremental reciprocal. The LUT size becomes therefore critical for R and for the choice of the machine precision. In fact, to hook the LUT entry backward, the machine precision has to be shoddy (<=) in order to get rid of this maximum approximation error because treated as machine error. Table 4:2 relates the LUT size with the maximum approximation error for the decremental reciprocal computed up to the second and the third order of approximation.

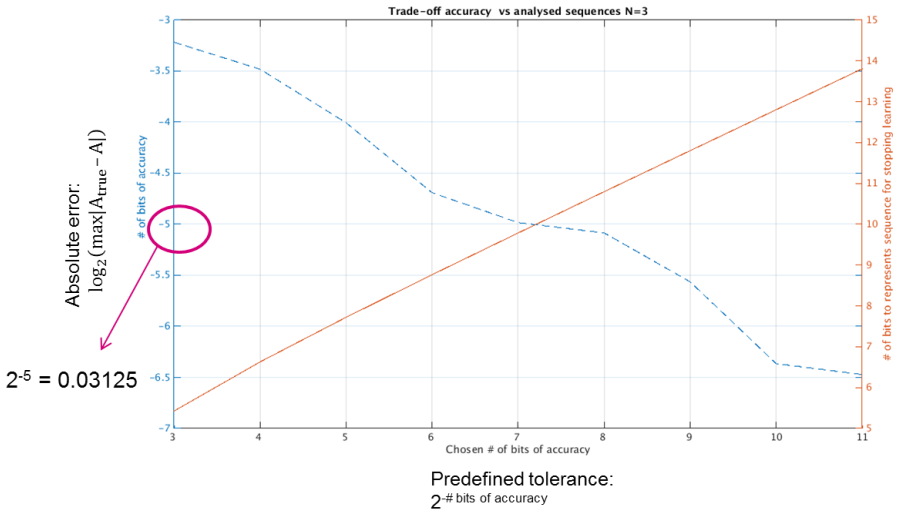
LUT size	Max Error 2nd degree	Equiv. in bits	Max Error 3rd degree	Equiv. in bits
4	2.7 e^{-2}	6	4.20 e^{-3}	8
8	4.0 e^{-3}	8	3.80 e^{-4}	12
16	8.0 e^{-4}	11	4.00 e^{-5}	15
32	2.0 e^{-4}	13	4.80 e^{-6}	18
64	4.7 e^{-5}	15	5.80 e^{-7}	21
128	1.2 e^{-5}	17	7.15 e^{-8}	24

Table 4:2 LUT size and the related maximum approximation error for both 2nd and 3rd approximation degree.

For instance, a LUT size of 32 entries requires a machine precision less or equal than 13 bits. Instead, by considering the third order approximation for the computation of the reciprocal with our method, $R = O(x^4)$ and the machine precision relaxes to less or equal than 18 bits. Bigger LUT size leads both an area and time overheads, while smaller LUT size reduces machine precision to such an extent our method approximation is no more tolerable.

4.3.3 Machine precision

The classification accuracy of the proposed algorithm increases with the presentation of new sequences because of the on-line elaboration. However, the learning stops when opportune stopping criteria are met. In particular, when the log-likelihood does not increase anymore or the parameter estimation does not improve below a predefined threshold.



© 2017 Tous droits réservés.

Thèse de Matteo Causo, Lille 1, 2017

Figure 4:8 Absolute parameter estimation error and number of analyzed sequences vs. predefined tolerance.

The plot in Figure 4:8, relates the predefined threshold with the effective absolute error between the learnt parameters and the true ones, for the synthetic application in Section 3.7.1, where the hidden model is known. If we accept an absolute error in the order of $O(2^{-5})$, the learning has to continue until the difference between two successive parameters estimations is strictly less than 2^{-7}. Once the predefined tolerance fixed, we can approximately estimate the number of successive sequences of input data to elaborate before the learning stops. For the synthetic application with $N = 3$, it results of about $2^{10} \approx 1000$ sequences, while for the ten MNIST handwritten digits recognition, around 2700-3000 sequences are required.

If we assume the learning stops after 5000 sequences, then the smallest reciprocal we need to compute in a bad worst case is for $a = 5000$ and results $x_{5000} = 1/5000 \approx 2^{-12.3}$. Therefore, we need a fractional part of 13 bits to correctly represent this number in a fixed-point representation, i.e. UQ.13 in the Q-format[13]. However, our method computes the square and so a UQ.25 representation is needed to result without approximation error because $x_{5000} = x_{4999} - x_{4999}^2 \approx 2^{-12.3} - 2^{-24.6}$.

For what concerns the digamma values representation, the smallest digamma increment will be $\psi(5000) - \psi(4999) = 2e^{-4} \approx 2^{-12.3}$, thus we need a UQ.13 for representing its fractional part. Conversely, the biggest representable range is $[\psi(1), \psi(5000)] = [-0.58, 8.52 \approx 2^4]$. Finally, we need a SQ4.13-18 for representing all possible values.

The choice of the machine precision is then affected by the maximum tolerated error in Table 4:2 and so indirectly by the LUT size; the precision for the reciprocal computation and the representation for all possible digamma values. Although the unitary part can be chosen to be SQ4.f, the fractional part requires a trade-off: it has to be less or equal than the maximum tolerated error in Table 4:2 in order to treat it as machine error, but at the same time has to be of 25 bits for the reciprocal to be properly computed.

For what concerns this latter, we are already tolerating a coarser approximation included into the R term. Therefore, a good compromise is to choose a machine precision of 20 bits so that the digamma representation results SQ4.15-20; the LUT size of 64 entries and reciprocal computation can be performed unsigned on 20 bits.

4.3.4 Log-domain FFBS

A further contribution of the present PhD thesis work is the simplification of the computations to perform the Forward Filtering Backward Smoothing (FFBS) algorithm in the Log-domain.

$$
\begin{aligned}
\log\left(\boldsymbol{A}^{*'}\boldsymbol{B}^{*}\right) &= \log\left(\begin{bmatrix} a_{11}^{*} & \dots & a_{N1}^{*} \\ \vdots & \ddots & \vdots \\ a_{1N}^{*} & \cdots & a_{NN}^{*} \end{bmatrix}\begin{bmatrix} b_{1}^{*} \\ \vdots \\ b_{N}^{*} \end{bmatrix}\right) \\
&= \log\left(\begin{bmatrix} a_{11}^{*}b_{1}^{*} + \cdots + a_{N1}^{*}b_{N}^{*} \\ \vdots \\ a_{1N}^{*}b_{1}^{*} + \cdots + a_{NN}^{*}b_{N}^{*} \end{bmatrix}\right) \\
&= \log\left(\begin{bmatrix} e^{\left(a_{11_{Log}}^{*} + b_{1_{Log}}^{*}\right)} + \cdots + e^{\left(a_{N1_{Log}}^{*} + b_{N_{Log}}^{*}\right)} \\ \vdots \\ e^{\left(a_{1N_{Log}}^{*} + b_{1_{Log}}^{*}\right)} + \cdots + e^{\left(a_{NN_{Log}}^{*} + b_{N_{Log}}^{*}\right)} \end{bmatrix}\right) \\
&= \begin{bmatrix} \log\left(\sum_{n=1}^{N} e^{\left(a_{n1_{Log}}^{*} + b_{n_{Log}}^{*}\right)}\right) \\ \vdots \\ \log\left(\sum_{n=1}^{N} e^{\left(a_{nN_{Log}}^{*} + b_{n_{Log}}^{*}\right)}\right) \end{bmatrix} \\
&\cong \begin{bmatrix} \max\left\{\left(a_{11_{Log}}^{*} + b_{1_{Log}}^{*}\right), \dots, \left(a_{N1_{Log}}^{*} + b_{N_{Log}}^{*}\right)\right\} \\ \vdots \\ \max\left\{\left(a_{1N_{Log}}^{*} + b_{1_{Log}}^{*}\right), \dots, \left(a_{NN_{Log}}^{*} + b_{N_{Log}}^{*}\right)\right\} \end{bmatrix}
\end{aligned}
$$

Equation 4:5 – Log-domain matrix multiplication.

According to Figure 4:2, $\boldsymbol{W}^{A}$ and $\boldsymbol{W}^{B}$ are frequencies counts, which the digamma function is applied to and they result in $\boldsymbol{A}_{Log}^{*}$ and $\boldsymbol{B}_{Log}^{*}$ respectively. In order to obtain $\boldsymbol{A}^{*}$ and $\boldsymbol{B}^{*}$, we have to apply a Log-to-Normal domain conversion, i.e. the element-wise

[13] The Q-format fixed-point representation we employ in the present PhD thesis work is: "**U/SQu.f-w"** where "**U/S**" refers to Un/Signed; "**u.**" is the number of bit for the unitary part; "**.f**" is the number of bit for the fractional part and "**w**" is the overall wordlength.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

exponential function. Our observation is that we can avoid this conversion if FFBS is reformulated to perform in the Log-domain. In particular, Equation 3:9 stays the same, while Equation 3:6 requires a bit more effort. Although, elementwise products and divisions trivially and conveniently become additions and subtractions in the Log-domain, matrix multiplications challenge for an innovating and valuable contribution.

For example, the matrix multiplication $\boldsymbol{A}^{*\prime}\boldsymbol{B}^{*}$ in Eq. 3.6, where $\boldsymbol{A}^{*} = e^{(\boldsymbol{A}^{*}_{Log})}$ and $\boldsymbol{B}^{*} = e^{(\boldsymbol{B}^{*}_{Log})}$, results in Log-domain as in Equation 4:5. Therefore, we employed the *log-sum-exp* (LSE) function that for large positive arguments is a smooth approximation of the maximum function[14]. Like multiplication operations become simple additions in Log-domain; addition operations become the LSE in the Log-domain and, in our case, the matrix multiplication becomes the maximum of simple additions in Log-domain.

4.3.5 Datapath (DP) and Control Units (CUs)

After all the optimizations introduced so far, the Datapath (DP) results very simple and consists of few operators, as in Figure 4:9.

The Max-Add-Sub (MAS) binary operator receives as input two operands and the selector for the operation to perform among Addition, Subtraction and choosing the Maximum. In this particular case, it outputs the value and the position of the larger operand between the two in input. All computations and comparisons are performed on 20 bits and they require a clock cycle. Two MAS operators are instantiated.

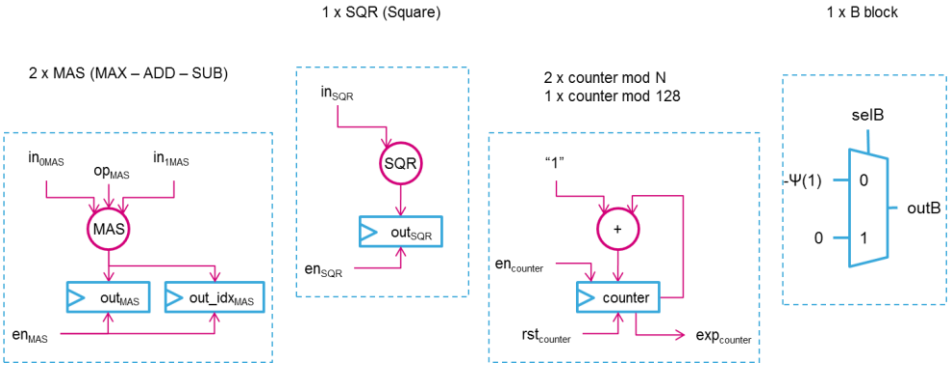


Figure 4:9 Operators in the datapath (DP).

The square (SQR) unary operator computes the square of the input in a clock cycle. The result is on 40 bits truncated to the 20 most significant ones. In fact, this operator is only exploited for the computation of the reciprocal and so it handles fractional numbers.

The counter block, if enabled, increments the value stored into a dedicated register at each clock cycle. Once it reaches expiration, it issues an acknowledge signal. It is instantiated three times: twice to scroll through bi-dimensional data structures and once for a system level counter to regularly trigger HMM pruning and merging operations.

Finally the multiplexer (MUX) block assigns each element of $\boldsymbol{B}^{*}$ with its initial value according to $\boldsymbol{Z}$, which is hardcoded to *selB*.

All the blocks are described in a fully parameterized Verilog code, so that their definition fits any architecture and they can be easily reused.

The control logic consists of five distributed Control Units (CUs) that autonomously orchestrate the overall elaboration by means of a dedicated handshaking protocol. In fact, instead of a complex centralized CU, we privileged a few small and simple CUs that elaborate local data and strive to use the DP without a global arbitration. We took inspiration from Globally Asynchronous Locally Synchronous (GALS) systems. However, in our case, the system clock is the same and is shared to all the CUs. The paramount advantage of asynchronous handshaking is that it simplifies the sequential logic and allows each CU to have a dynamic processing time that fits to the workload. This is critical in a highly dynamic system where the overall complexity changes on-the-fly, while learning the probabilistic model structure and parameters.

Memory is distributed as well. In fact, a key principle of the Bayesian Machine Learning (BML) is to extract the knowledge from data and to store and elaborate only a few parameters and important variables. Fetching data from an external memory is a rare event

[14] Log-Sum-Exp (LSE) Function and Properties.

© 2017 Tous droits réservés.

and it is delegated to the uP executing the VBEM GMM or VBE HMM GMM. In fact, in the FB + VBEM HMM CoP we elaborate high level knowledge and we benefit of local dedicated Register Files (RFs). Therefore, we build on an elaboration that is local and very close to the memory.

CU	RF	DP	Goal
Digamma	$DigW^A$ $RecW^A$ DigAccum RecAccum	MAS_0 MAS_1 SQR_0	Computes Digamma (Dig) and Reciprocal (Rec) for both W^A and Accum
Fil	w	$counter_0$ MAS_0 MAS_1	Computes filtered probabilities during FFBS
GFP	A* A*_idx	$counter_0$ $counter_1$ MAS_0 MAS_1	Greedy Forward Pass. It takes trace of optimized values in A*_idx.
Add	every	$counter_0$ $counter_1$ MAS_0 MAS_1 SQR_0	Adds a new HMM state on-the-fly and modifies the model structure
Prune	every	$counter_0$ $counter_1$ MAS_0 SQR_0	Prunes the least used HMM state on-the-fly.

Table 4:3 Summary of CUs goals, dedicated Register Files and employed DP operators.

Table 4:3 describes each CU and details its goal, its dedicated RF and the DP operators it employs, while Figure 4:10 gives a general idea of the conceived system.

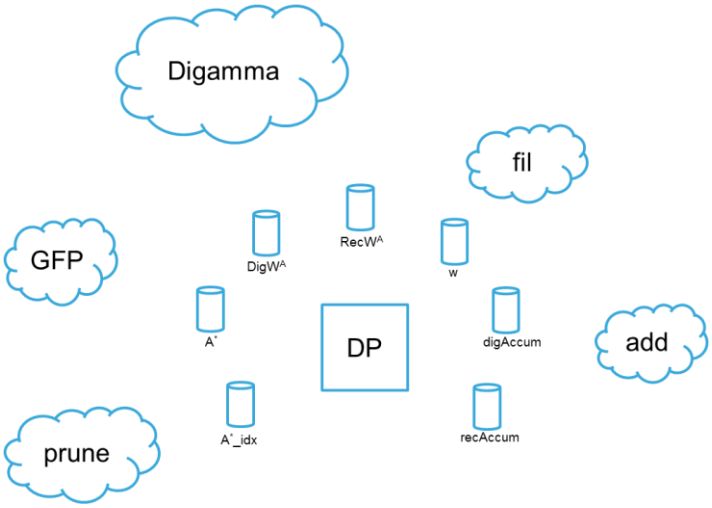


Figure 4:10 Overall illustrative idea of the conceived CoP.

(The representation follows the following convention: cloud = Control Unit (CU), cylinder = Register File (RF), square = operators.)

According to Figure 4:10, the DP is centralized and exploited by each CU in a time-multiplexed scheme. It is general enough to fulfill all possible requirements, but it is still dedicated and essential because properly dimensioned for the few tasks it is asked to perform. In fact, its definition follows the Ultra-Low-Power (ULP) design philosophy rather to strive for reaching high-performance by employing additional resources. Nevertheless, it is fully compliant with all possible applications we are targeting. In particular, medical applications require processing the biosignals summarized in Table 4:4 [Bemmel and Musen 1997], which are sampled at very low frequencies. For instance, Electroencephalogram (EEG) signals are generally sampled at 256 Hz.

Signal	*Bandwidth (Hz)*	*Amplitude range*
Electroencephalogram (EEG)	0.2 - 50	600 µV
Electrooculogram (EOG)	0.2 - 15	10 mV

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Electrocardiogram (ECG)	0.15 - 150	10 mV
Electromyogram (EMG)	20 - 8000	10 mV
Blood pressure	0 - 60	400 mm Hg
Spirogram	0 - 40	10 L
Phonocardiogram (PCG)	5 - 2000	80 dB

Table 4:4 Bandwidths and Amplitude Ranges of some frequently used biosignals.

4.3.6 Synthesis and implementation

The FB + VBEM HMM CoP has been synthetized and mapped in the Xilinx Zynq®-7020 All Programmable SoC embedded into a ZedBoard, i.e. the evaluation board from Avnet. Vivado Design Suite 2015.4 has been employed.

The Xilinx Zynq®-7020 All Programmable SoC embeds a reduced but functional FPGA with many dedicated, full custom, low-power DSP slices (DPS48E). Implementation results on the FPGA platform are summarized in Table 4:4. In particular, a DSP48E is employed for the SQR block in the DP.

The FB + VBEM HMM CoP has also been synthetized for the 28ST-FDSOI CMOS technology through the standard digital flow and results are summarized in Table 4:4. Compared to its FPGA implementation, the dedicated ASIC has almost the same number of registers but a smaller number of combinational logic units. This is due to the overhead for the flexibility and programmability of the FPGA. In fact, the equivalent combinational logic for the 12k LUTs is around 185k, while the dedicated ASIC requires only 25k combinational logic units in total, which is comprehensive of the SQR block, as well. Therefore, there is a further interesting gain in power consumption that translates into 23x reduction in energy per input sequence classification.

		Zynq®-7020	28ST-FDSOI @ 0.6V
Power consumption @ 50 MHz		8 mW	351 uW
Energy per seq. classification		160 pJ	7 pJ
Area		-	0.0264 mm^2
Logic utilization	LUT	12,127	-
	Registers	5,261	6,073
	Block RAM	0	-
	DSP48E	1	-
	Combinational	185,471 (Equivalent)	25,179

Table 4:5 HW implementation results for the FB + VBEM HMM CoP.

(The FB + VBEM HMM CoP has been mapped into the Zynq®-7020 and synthetized for 28ST-FDSOI @ 0.6V. Power consumption is estimated from simulated switching activity for both implementations and energy per sequence classification is derived.)

The comparison with state-of-the-art solutions in Table 4:1 is unfair, because of the different technologies, approaches and applications. However, it allows getting an idea of where to place our work in the state-of-the-art and to validate our choices and optimizations strategies. Therefore:

- The FB + VBEM HMM CoP like the Spatial Pooler for HTM implemented in [Zyarah and Kudithipudi 2016] is only one block of a more complete system. However, our implementation inspired by HTM requires half the number of LUTs and no Block RAM, but a limited number of registers that is always much smaller than the 52 block RAMs used in [Zyarah and Kudithipudi 2016].

- The FB + VBEM HMM CoP completely implements the learning of the HMM through both estimation of HMM parameters and inference through Forward Filtering Backward Smoothing (FFBS) procedures. Therefore, by comparing its implementation results with the approaches in [Biradar et al. 2015; Ren et al. 2014], we can actually validate our choice for the Bayesian Machine Learning (BML) with respect to Forward Neural Networks and kernel methods, respectively. In fact, as expected both logic and memory requirements are widely relaxed in our method because the Bayesian framework is inherently less complex. Furthermore, it allowed for both algorithm level and HW implementation optimizations that actually lowered the memory wall effect and allowed for device variability through the probabilistic computing.

- The FB + VBEM HMM CoP compared with the implementation in [Jewajinda and Chongstitvatana 2012b] still requires a much smaller computing power. No comparison is possible with the approach in [Sasagawa and Mori 2016] because they targeted very high performance and they were no limited in power consumption. In fact, their budget is of 15W. Instead, the low-power implementations in [Park et al. 2015; Benatti et al. 2015] register higher power consumption with respect to our approach even if they do not implement learning neither on-line nor in-situ. Thus, we show the BML is a better

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

choice for IoT and medical application, because it performs generally better than dedicated accelerator for Deep Learning and supervised Supporting Vector Machines.

Finally, we demonstrated the BML leads to convenient and attractive optimizations even without the need to migrate towards a different technology than CMOS.

4.4 FPGA demos

The overall SoC introduced in Section 4.2 has been tested systematically from its software-based simulation to its implementation in the FPGA platforms. Therefore, several demos have been realized to evaluate its performance, especially with respect to the test applications in Section 3.7.

In particular, the proposed on-line VBEM HMM GMM algorithm has been implemented and evaluated in MATLAB. Then, the SoC architecture in Section 4.2 has been obtained by rearranging the MATLAB code and separating each task into a specific block. The FB + VBEM HMM CoP has been then studied, optimized and described in both SystemVerilog and Verilog HDL. It has been successively synthetized and implemented for the Xilinx Zynq®-7020 All Programmable SoC through the Vivado Design Suite 2015.4. The FPGA in Xilinx Zynq®-7020 has been then programmed accordingly.

To test the FB + VBEM HMM CoP independently of the rest of the proposed SoC, we record the input and output of the MATLAB routine simulating the FB + VBEM HMM CoP and define a test bench for its implementation. In particular, input data are sent to the FPGA and outputs are collected from it through JTAG for the synthetic application in Section 3.7.1. A comparison between collected results and exact recorded ones validated the proper functionality of our implemented CoP.

To automatize the tests and to connect the MATLAB code directly with the implemented block, we exploit the uProcessor provided into the Xilinx Zynq®-7020 and we result with three different demos of increasing complexity.

In fact, the Xilinx Zynq®-7020 is a ductile IC that consists mainly of two blocks internally connected: the Programmable Logic (PL), i.e. the FPGA and the Processing System (PS), i.e. the uProcessor. This latter is actually a powerful Dual ARM® CortexTM-A9 MPCore™ with CoreSight™, so it is able to run a dedicated distribution of the Linux Operating System.

Therefore, we manipulated the Linux distribution and wrote an executable for the PS allowing it to access the STMicroelectronics (ST) intranet though Ethernet and to act as a TCP Server. In particular, once connected to the ST intranet, the PS is assigned with a specific IP address and thus is reachable from the MATLAB code running on the ST servers and acting as TCP Client. In other words, the MATLAB code we wrote for the proposed on-line VBEM HMM GMM accesses to the ST intranet, remotely connects to the PS of the Xilinx Zynq®-7020 and exchanges with it data by using the TCP protocol. This setup is resumed in Figure 4:11.

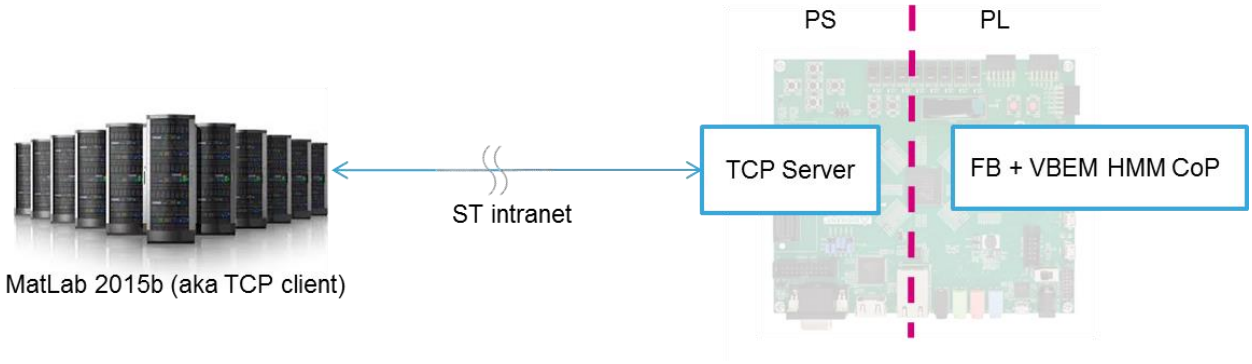


Figure 4:11 MATLAB remotely connected to the ZedBoard for testing purpose.

The TCP Server application running on the PS of the Xilinx Zynq®-7020, manages the data transfers between the MATLAB TCP Client and the FB + VBEM HMM CoP by using an elaborated mechanism of both HW and SW interrupts to synchronize all parts involved and maintain real-time elaboration performance.

© 2017 Tous droits réservés.

4.4.1 Demo 1

We firstly tested the synthetic application in Section 3.7.1, in order to validate the setup in Figure 4:11. In particular, the MATLAB code retrieves inputs, applies the preprocessing by simulating the Feature Extraction Engine (FEE) and controls the overall elaboration through the Control (Ctr) block. This latter, is in charge of establishing and maintaining the communication with the PS; of analyzing and evaluating classification results and of outputting them in a user-friendly interface. The uP block in Figure 4:12 incorporates all the SoC blocks in Figure 4:1 that are maintained software-based, while the CoP block represents the FPGA implementation of the FB + VBEM HMM CoP.

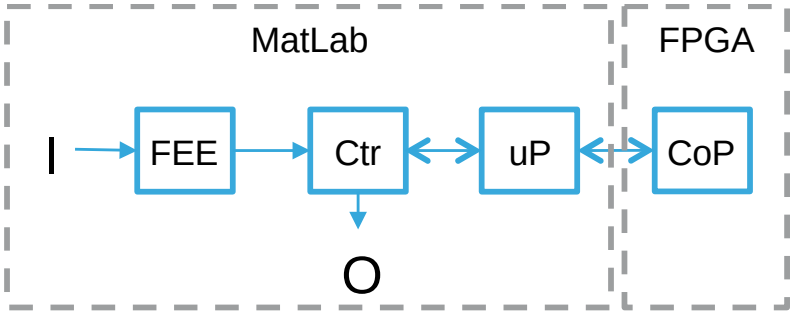


Figure 4:12 Schematic representation of the system implemented for Demo 1.

4.4.2 Demo 2

In the Demo 2 represented in Figure 4:13, we build the SoC proposed in Section 4.2 by making it independent on its MATLAB implementation. In this way, we are able to estimate the overall system performance in terms of execution time and power consumption. Moreover, we can test the system on a continuous stream of input data as in a real-world IoT or medical application.

We force the uP block in Figure 4:12 to move on the ZedBoard and to be executed by its PS. Therefore, we convert the MATLAB code into ANSI C code compiled for the ARM® CortexTM-A9 in the PS.

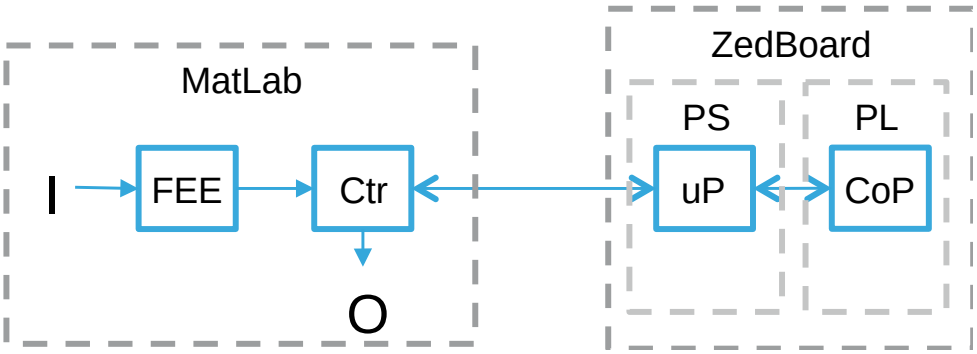


Figure 4:13 Schematic representation of the system implemented for Demo 2.

4.4.3 Demo 3

The Xilinx Zynq®-7020 embeds a Dual ARM® CortexTM-A9. Thus, it has two cores that can be employed independently to perform as single nodes in a distributed system as shown in Figure 4:14. We validated in this way the distribution results in Section 3.7.2.

© 2017 Tous droits réservés.

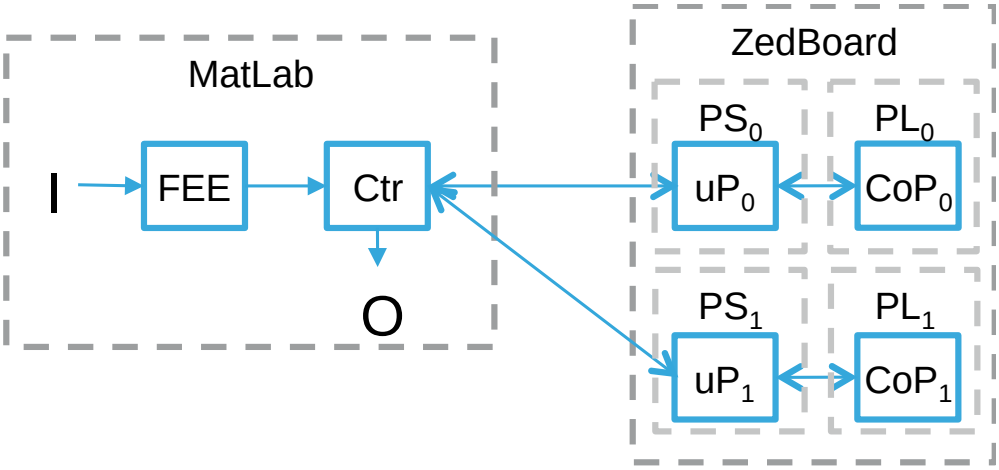


Figure 4:14 Schematic representation of the system implemented for Demo 3.

4.5 Conclusions

According to [HALO 2015] an acceleration factor of 10^2 -10^4 is critically needed to achieve orders of magnitude improvement in performance, energy efficiency and compactness with respect to traditional ML software-based on CPUs/GPUs.

In Chapter 3, we have already demonstrated two orders of magnitude in complexity reduction with only algorithm level considerations and manipulations on the state-of-the-art VBEM HMM GMM. We regularly emphasized those optimizations aimed to a feasible and attractive HW implementation of the proposed on-line VBEM HMM GMM. In this Chapter, we analyzed the state-of-the-art for Learning-On-a-Chip solutions summarized in Table 4:1 and we identified the margin for a further order of magnitude in complexity reduction through conventional HW optimization techniques. Therefore, we built a proof-of-concept on a FPGA platform to demonstrate the main idea behind this PhD attains our main objective.

The achievements of this Chapter are the following:

- We validate our choice for the BML by demonstrating it leads to convenient and attractive optimizations even without the need to migrate towards a different technology than the CMOS. In fact, even if through unfair comparisons, we can state it is more suitable than other Neuro-Inspired approaches to Learning-On-a-Chip solutions such as Forward Neural Networks and kernel methods. Furthermore, BML is a better choice for IoT and medical application, because it performs generally better than dedicated accelerator for Deep Learning and supervised Supporting Vector Machines.

- We defined a dedicated solution for the digamma function $\psi(\cdot)$ that is hard to compute analytically and it is generally approximated even in software-based implementations. We conveniently exploited the on-line elaboration and defined an incremental/decremental digamma calculator. With this same principle, we built on an incremental/decremental reciprocal calculator useful for the digamma block. Both contributions represent an innovation with respect to the state-of-the-art.

- We further optimized the FFBS algorithm by allowing it to perform in Log-domain. Therefore, we employed the log-sum-exp (LSE) function to approximate linear addictions in the Log-domain and we result with an optimized and sophisticated matrix multiplication technique.

- We finally built on an advanced setup to flexibly test our HW implementation on a FPGA platform by means of three real-time demos on the same applications of interest as in Section 3.7.

Next Chapter opens toward further contributions made in the framework of this PhD thesis, which are related but not fundamental to this thesis main objectives.

© 2017 Tous droits réservés.

Thèse de Matteo Causo, Lille 1, 2017

References

JH Van Bemmel and M.A. Musen. 1997. Handbook of Medical Informatics. *Handb. Med. Informatics* 362 (1997), 621.

Simone Benatti et al. 2015. A Versatile Embedded Platform for EMG Acquisition and Gesture Recognition. *IEEE Trans. Biomed. Circuits Syst.* 9, 5 (October 2015), 620–630. DOI:http://dx.doi.org/10.1109/TBCAS.2015.2476555

Ravikant G. Biradar, Abhishek Chatterjee, Prabhakar Mishra, and Koshy George. 2015. FPGA Implementation of a Multilayer Artificial Neural Network using System-on-Chip Design Methodology. (2015), 1–6. DOI:http://dx.doi.org/10.1109/CCIP.2015.7100683

HALO. 2015. IEEE / ACM Workshop on Hardware and Algorithms for Learning On-a-chip (HALO) Summary Report. (2015).

Yutana Jewajinda and Prabhas Chongstitvatana. 2012a. A parallel genetic algorithm for adaptive hardware and its application to ECG signal classification. *Neural Comput. Appl.* 22, 7–8 (2012), 1609–1626. DOI:http://dx.doi.org/10.1007/s00521-012-0963-9

Yutana Jewajinda and Prabhas Chongstitvatana. 2012b. A parallel genetic algorithm for adaptive hardware and its application to ECG signal classification. *Neural Comput. Appl.* 22, 7–8 (2012), 1609–1626. DOI:http://dx.doi.org/10.1007/s00521-012-0963-9

Shihao Ji, Balaji Krishnapuram, and Lawrence Carin. 2006. Variational Bayes for continuous hidden Markov models and its application to active learning. *IEEE Trans. Pattern Anal. Mach. Intell.* 28, 4 (2006), 522–532. DOI:http://dx.doi.org/10.1109/TPAMI.2006.85

Gopinath Mahale, Soumitra K. Nandy, Eshan Bhatia, and Ranjani Narayan. 2016. VOP: Architecture of a Processor for Vector Operations in On-Line Learning of Neural Networks. In *Proceedings of the IEEE International Conference on VLSI Design*. 391–396. DOI:http://dx.doi.org/10.1109/VLSID.2016.65

Niels Möller and Torbjörn Granlund. 2011. Improved division by invariant integers. *IEEE Trans. Comput.* 60, 2 (2011), 165–175. DOI:http://dx.doi.org/10.1109/TC.2010.143

Seong-Wook Park et al. 2016. An Energy-Efficient and Scalable Deep Learning/Inference Processor With Tetra-Parallel MIMD Architecture for Big Data Applications. *IEEE Trans. Biomed. Circuits Syst.* 9, 6 (2016), 1–1. DOI:http://dx.doi.org/10.1109/TBCAS.2015.2504563

Seong-wook Park, Junyoung Park, Student Member, and Kyeongryeol Bong. 2015. An Energy-Efficient and Scalable Deep Learning / Inference Processor With Tetra-Parallel MIMD Architecture for Big Data Applications. 9, 6 (2015), 838–848.

Xiaowei Ren, Pengju Ren, Badong Chen, Tai Min, and Nanning Zheng. 2014. Hardware implementation of KLMS algorithm using FPGA. In *2014 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2276–2281. DOI:http://dx.doi.org/10.1109/IJCNN.2014.6889689

Yukihiro Sasagawa and Atsuhiro Mori. 2016. High-level Video Analytics PC Subsystem Using SoC With Heterogeneous Multicore Architecture. *IEEE J. Solid-State Circuits* 51, 4 (April 2016), 1051–1059. DOI:http://dx.doi.org/10.1109/JSSC.2015.2501984

Zisheng Zhang and Keshab K. Parhi. 2016. Low-Complexity Seizure Prediction From iEEG/sEEG Using Spectral Power and Ratios of Spectral Power. *IEEE Trans. Biomed. Circuits Syst.* 10, 3 (June 2016), 693–706. DOI:http://dx.doi.org/10.1109/TBCAS.2015.2477264

Abdullah M. Zyarah and Dhireesha Kudithipudi. 2016. Reconfigurable hardware architecture of the spatial pooler for hierarchical temporal memory. *Int. Syst. Chip Conf.* 2016–Febru (2016), 143–148. DOI:http://dx.doi.org/10.1109/SOCC.2015.7406930

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Chapter 5 Further contributions

Apart from the main objective that led to the contributions of this PhD thesis described so far, further studies and investigations in the field of Bayesian Machine Learning (BML) flow into further contributions proposed in this Chapter. In particular, inference mechanisms on arbitrary cyclic Probabilistic Graphical Model (PGM) are studied in detail and few criteria for exact inference are derived. The Expectation-Maximization (EM) algorithm is improved by performing inference during the E-step and a novel Ultra-Low-Power (ULP) feature extraction technique for biosignals is introduced. Finally, the doctoral experience was enriched with the mentoring of three internships on related topics.

5.1 Exact inference on arbitrary cyclic graphs

Graphical models define an intuitive framework and a clean mathematical formalism to represent and manipulate joint probability distributions [Jordan and Weiss 1996].

Probabilistic inference over graphical models aims to solve two classes of NP-hard [Cooper 1990; Shimony 1994] problems, namely marginals computation and Maximum A Posteriori (MAP) estimation, detailed in Section 5.1.1.

An efficient technique to solve both these problems is to exploit Belief Propagation (BP) algorithms, i.e. local concurrent message passing schemes grouped and unified according to the well-known Generalized Distributive Law (GDL) [Aji and Mceliece 2000]. Therefore, marginals computation could be solved by the sum-product algorithm, referred as “belief update” and MAP estimation by the max-product algorithm, also called “belief revision” [Weiss 1997].

Theoretically, these algorithms are guaranteed to converge and to result with exact inference only for singly connected graphs [Pearl 1988]. For arbitrary cyclic graphs only approximate inference can be performed, and the brute application of BP is hence referred to as Loopy BP.

Surprisingly, Loopy BP produces excellent results when applied in practice to both for marginal computations and MAP estimation tasks: error correcting codes, speech recognition, image understanding, stereo matching, photomontage, background segmentation, and de-noising in computer vision are excellent example of those. For instance, the detection of Turbo Codes [Berrou et al. 1993] shocked the coding-theory community with its truly excellent results.

In essence, all of these built on key intuition conjectured by [McEliece et al. 1998]: “There are general undiscovered theorems about the performance of BP on loopy Direct Acyclic Graph (DAG)s. These theorems [...] will show that [...] BP “converges with high probability to a near-optimum value” of the desired belief on a class of loopy DAGs.”

This intuition triggers two basic questions:

1. “Is there something special about the error- correcting code setting, or does Loopy BP work as an approximation scheme for a wider range of networks?” [Murphy et al. 1999]

2. Are there any theoretical insights that govern the behavior of Loopy BP?

The first question was addressed in [Murphy et al. 1999]. To the best of our knowledge, the second is still unanswered, because “a theoretical understanding of the algorithms behavior in such [loopy] networks has yet to be achieved” [Weiss 1997].

Initial attempts focused on manipulating the graph structure [Weiss and Freeman 2001] to gain insights on the behavior of particularly regular graphs, such as the single loop and “balanced” graphs concepts [Weiss 1997]. Subsequently, the focus has shifted to understanding how to guarantee and improve convergence in arbitrary cyclic graphs by decomposing them into several singly connected trees and by performing exact inference on them.

This approach, known as the Tree-ReWeighting or Tree-ReParametrization (TRW) algorithm [Wainwright et al. 2003], was substantially improved in [Kolmogorov 2006], which proposed a sequential scheduling for the message passing to better guarantee

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Matteo Causo, Lille 1, 2017

convergence (TRW-S). Finally, [Meltzer et al. 2009] presents these algorithms as instances of a unified framework for convergent message passing algorithms by enforcing some consistency conditions.

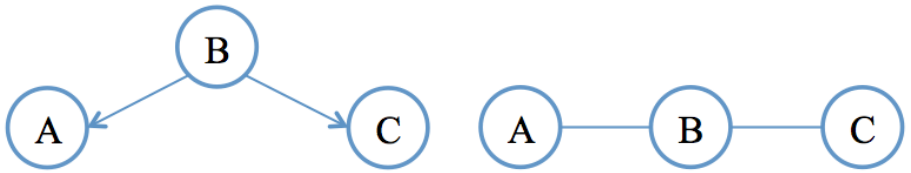
Recently, the growing interest in machine-learning and data science has triggered the hardware community to pay increasing attention to special architectures to improve performance and efficiency for this class of problems. However, the metrics pursued by computer architects are slightly different from those conventionally targeted by computer science. In addition to the algorithm's performance, execution time, computational cost, energy-efficiency and the physical size of the final hardware are critical. For these reasons, the basic BP algorithm, being concurrent, distributed and locally executed, is preferred, for instance, to the TRW-S, which requires a sequential execution and a higher-level centralized management of the spanning trees scheduling.

Here, we intend to provide an answer to the second question by revisiting the basic algorithms for BP. We analyze the mechanism that enables exact inference on singly connected graphs with binary variables. In particular, we focus on the singularity of messages rather than on the graph structure. We derive that what is of paramount importance is the management of the redundancy meant as both the quantity and the quality of the messages received for each node. Indeed, we justify the presence of oscillations and we figure out how many iterations are necessary and sufficient to converge with exact inference. In Section 5.1.6, we apply our findings to the case of graphs with a single loop. In Section 5.1.7, we perform exact inference on the toy Turbo Codes proposed in [Weiss 1997] as an example of a "balanced" graph. We generalize our approach for arbitrary cyclic graphs in Section 5.1.8 and finally, in Section 5.1.9, we conclude and discuss further generalizations for dynamically changing networks as demanded in Multi-Agent Systems (MAS) real-word applications.

5.1.1 Exact inference

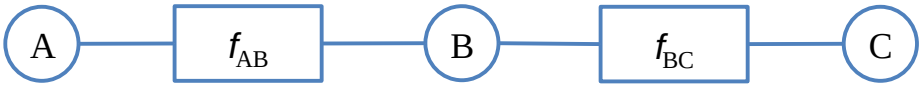
A graphical model consists of a graph composed of nodes and edges. Nodes represent Random Variables (RVs) and edges qualitative dependencies between them. The quantitative dependences are defined by parameterized conditional distributions, generally referred as parameters of the graphs or as "potential functions". The pattern of all edges defines the structure of the graph. Both structure and parameters specify the joint probability distribution over all variables in the graph [Ghahramani 2002].

There are several types of graphs underlying a graphical model, but the usual classification distinguishes between Direct Acyclic Graph (DAG) and Undirected Cyclic Graph (UCG). Examples are, respectively, Bayesian Network (BN) in Figure 5:1a and Markov Random Field (MRF) in Figure 5:1b.



(a) Bayesian Network

(b) Markov Random Field



(c) Factor graph

Figure 5:1. Examples of graphical models.

The joint probability distribution over the three binary RVs A, B, C for the MRF in Figure 5:1b, is exhaustively represented in Table 5:1 and is expressed as:

$$\mathrm{P}(\mathrm{A},\mathrm{B},\mathrm{C}) = \frac{1}{\mathrm{Z}}[\mathrm{P}(\mathrm{A})\psi_{\mathrm{AB}}(\mathrm{A},\mathrm{B})\mathrm{P}(\mathrm{B})\psi_{\mathrm{BC}}(\mathrm{B},\mathrm{C})\mathrm{P}(\mathrm{C})]$$

Equation 5:1 – Joint probability distribution for the MRF in Figure 5:1b.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

where $P(X)$ represents the prior on the belief for the RV X and $\psi_{XY}(X,Y)$ the potential function for the edge XY.

A	**B**	**C**	**P(A)**	**P(B)**	**P(C)**	$\boldsymbol{\psi_{AB}(A,B)}$	$\boldsymbol{\psi_{BC}(B,C)}$	$\boldsymbol{\tilde{P}(A,B,C)}$
$\boldsymbol{a_0}$	b_0	c_0	$P(A=a_0)$	$P(B=b_0)$	$P(C=c_0)$	$\psi_{AB}(A=a_0,B=b_0)$	$\psi_{BC}(B=b_0,C=c_0)$	$\tilde{P}(A=a_0,B=b_0,C=c_0)$
…	…	…	…	…	…	…	…	…
$\boldsymbol{a_1}$	b_1	c_1	$P(A=a_1)$	$P(B=b_1)$	$P(C=c_1)$	$\psi_{AB}(A=a_1,B=b_1)$	$\psi_{BC}(B=b_1,C=c_1)$	$\tilde{P}(A=a_1,B=b_1,C=c_1)$

Table 5:1 Unnormalized joint probability distribution table over the MRF and the FG in Figure 5:1.

Each row in Table 5:1 accounts for a different combination of the three binary RVs, for which the discrete unnormalized joint probability is computed by multiplying each column in the same row. Normalization consists in dividing every element of the $\tilde{\boldsymbol{P}}(\boldsymbol{A},\boldsymbol{B},\boldsymbol{C})$ column by Z, i.e. the sum over the whole column.

Generally, building this table is prohibitive because of its computational cost, which is exponential in the number of RVs.

As previously mentioned, probabilistic inference consists in manipulating the joint probability distribution for achieving two different tasks: marginals computation and MAP estimation. The former aims to compute the probability of a specific variable, for example A, to be in a particular state, $P(A=a_0)$ given all the observations:

$$\mathrm{P(A=a_0)} = \sum_{B,C} \mathrm{P(A=a_0,B,C)}$$

Equation 5:2 – Marginal computation by marginalization out of unobserved variables.

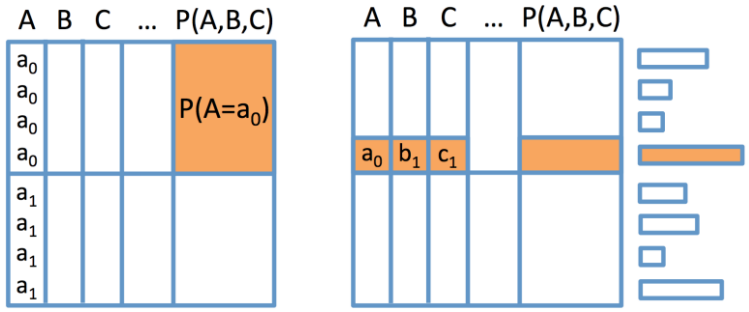
This operation is called marginalization out of the unobserved variables, i.e. B and C, and referring to Table 5:1, it consists in summing up together the discrete joint probabilities for all the rows accounting for $A=a_0$.

MAP estimation, instead, consists in finding the most likely sequence of states for the RVs that maximizes the joint probability distribution given all the evidences:

$$(\mathrm{a^*,b^*,c^*}) = \arg\max_{\mathrm{A,B,C}} P(A,B,C)$$

Equation 5:3 – MAP estimation by maximization.

Both operations performed over a joint probability distribution table, are schematized in Figure 5:2.



(a) Marginals (b) MAP estimation

Figure 5:2. Graphical example of probabilistic inference.

Despite its banality, this analysis unveils a key point that has been often underestimated and that results to be fundamental for performing exact inference: for both tasks of probabilistic inference, a complete knowledge about the overall graphical model (parameters, RVs and structure) is essential.

5.1.2 Belief Propagation

The joint probability distribution in Equation 5:1 can be expressed in a factored form schematized by the FG in Figure 5:1c:

$$P(A,B,C) = f_{AB}(A,B)\cdot f_{BC}(B,C)$$

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Matteo Causo, Lille 1, 2017

Equation 5:4 – Factor form for the joint probability distribution in Equation 5:1.

This suggests that the marginals computation Equation 5:2 and MAP estimation Equation 5:3 could be easily understood as a sum of products and as a max of products, respectively. Therefore, instead of computing inference by exhaustive enumeration, more efficient and interesting concurrent algorithms based on local message passing have been commonly exploited, according to the Generalized Distributive Law (GDL).

In particular, the sum-product algorithm, referred to as "belief update", provides an efficient local message passing procedure to compute the marginal functions simultaneously. At each iteration, every node computes outgoing messages and propagates them along the edges of the corresponding graphical model. Convergence is reached when for each node, input messages do not change anymore. At this point, the marginal belief is computed locally by multiplying among all the received messages. For the factor graphs encountered hereafter, these messages take two forms:

$$X \to f_{XY} = \prod_{f'_{XY} \in \mathcal{M}(X) \setminus f_{XY}} f'_{XY} \to X$$

Equation 5:5 – Message passing from RVs to factors.

$$f_{XY} \to X = \sum_{Y} [f_{XY}(X,Y) \cdot Y \to f_{XY}]$$

Equation 5:6 – Message passing from factors to RVs.

where X and Y are generic RVs, f_{XY} is a generic factor with scope {X, Y} and $\mathcal{M}(X)$ is the set of factors sending a message to X.

Similarly, the max-product, also known as "belief revision" can be derived from the sum-product by substituting the sum with the max operation. Therefore, Equation 5:6 modifies in:

$$f_{XY} \to X = \max_{Y} [f_{XY}(X,Y) \cdot Y \to f_{XY}]$$

Equation 5:7 – Message passing from factors to RVs.

Applying BP on singly connected graphs like the FG in Figure 5:1c is guaranteed to result with exact inference. In contrast, forcing BP on a cyclic graph will perform approximate inference, i.e. belief update results with generally wrong values, sometime completely discordant with the exact ones and generally belief revision does not converge. However, if it converges, it generally results with correct values [Weiss and Freeman 2001].

In next Subsection, we understand why BP is performing exact inference on singly connected graphs. Then, we examine what is causing Loopy BP to result with approximate inference on a single loop graph.

5.1.3 Singly connected graph example

Consider the FG in Figure 5:1c. We perform belief update by applying iteratively Equation 5:5 and Equation 5:6 for marginals computation. The resulting message passing dynamics are those shown in Figure 5:3. For convenience, at each iteration a complete set of $X \to f_{XY}$ and $f_{XY} \to Y$ messages is computed in order to directly move from X to Y, without loss of generality. Furthermore, the prior on the belief for each RVs is noted with the capital letter corresponding to the RVs, rather than using $P(\cdot)$.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

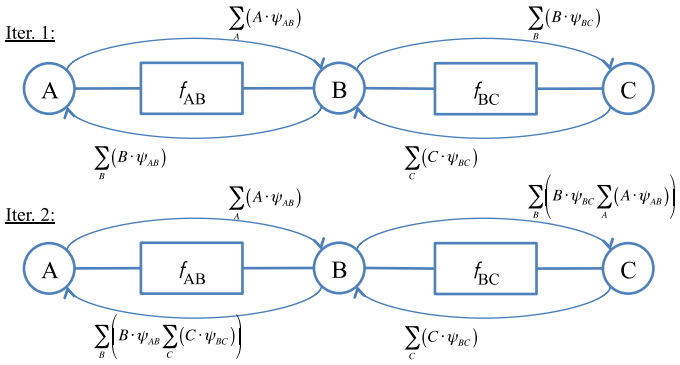


Figure 5:3. Exact marginals computation over a singly connected graph.

Convergence is reached in two iterations because for each node no input message is changing after that. This is due to the leaf nodes A and C, which transmit always the same message.

At this point, every RV node can compute its marginal by multiplying all incoming messages by its own prior belief. For instance, node A computes $P(A = a_0)$ as follows:

$$\mathrm{P_{i+1}(A = a_0) = P_i(A = a_0) \cdot \sum_B \left(P_i(B) \psi_{AB}(A = a_0, B) \cdot \sum_C P_i(C) \psi_{BC}(B, C) \right)}$$

Equation 5:8 – Marginal computation at the node A.

Expanding, Equation 5:8 becomes the sum of the discrete joint probabilities for all the rows in Table 5:1 that account for $A = a_0$ and so, yields Equation 5:2.

5.1.4 Matrix equivalent messages computation

A matrix equivalent computation for the messages is provided in Figure 5:4 and it is adopted hereafter. In particular, each RV node encodes its prior on the belief in a diagonal matrix and each potential function, ψ_{XY}, is represented by a square matrix, generally referred as "transition matrix". For the binary case, they result to be:

$$X = \begin{bmatrix} P(x_0) & 0 \\ 0 & P(x_1) \end{bmatrix}; \; \psi_{XY} = \begin{bmatrix} \psi(x_0, y_0) & \psi(x_1, y_0) \\ \psi(x_0, y_1) & \psi(x_1, y_1) \end{bmatrix}$$

Equation 5:9 – Matrix representation message passing.

A message traversing a node is combined with the node's own information by a matrix multiplication where the incoming message appears on the left side. For example the message going from A to B passing through f_{AB} in Figure 5:4, results to be $A\psi_{AB}$. Vice versa, the message going from B to A has to be multiplied by the transpose of ψ_{AB}, and so it results to be $B\psi_{AB}^T$. As visual convention adopted hereafter, each line in the message passing represents a further iteration with the following convention: the closer to the arrow, the less recent the message.

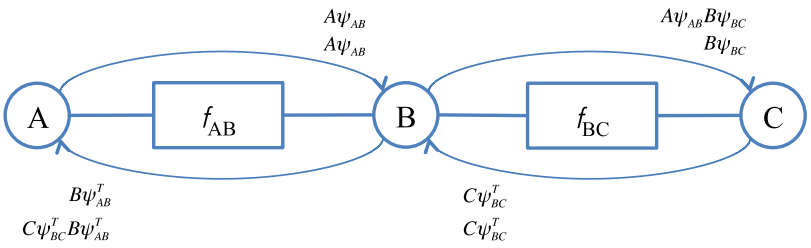


Figure 5:4. Matrix equivalent exact marginals computation.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Computing the marginals or the MAP estimate on each RV node, translates into an element-wise multiplication (Hadamard product, "∘") among all incoming messages for the node in order to combine them consistently. For instance, node B computes its marginal by means of Equation 5:10.

$$\mathrm{P}(\mathrm{B}=\mathrm{b}_0)=\sum_{i,1}\left[\left(\mathrm{A}\psi_{\mathrm{AB}}\circ\mathrm{C}\psi_{\mathrm{BC}}^{\mathrm{T}}\right)B\right]$$

Equation 5:10 – Marginal computation at the node B.

where i is the row index and the column related to b_0 is selected by the constant "1". For node A, Equation 5:8 simply becomes Equation 5:11, because only a single message is received.

$$\mathrm{P}(\mathrm{A}=\mathrm{a}_0)=\sum_{i,1}\left[\left(\mathrm{C}\psi_{\mathrm{BC}}^{\mathrm{T}}\mathrm{B}\psi_{\mathrm{AB}}^{\mathrm{T}}\right)A\right]$$

Equation 5:11 – Marginal computation at the node A.

5.1.5 Criteria for exact inference

Some key criteria could be defined from the observation of the previous generally valid example in order to guarantee exact inference.

Criterion 1: Each node needs to receive all other nodes' messages in order to have a complete global vision. In fact, even if the computation and messages are local, the information needed for performing exact inference has to come from everywhere in the graph.

Criterion 2: The consistency of the messages is enforced by the graph structure. In fact, referring to Figure 5:4, the message sent by the RV C, is enriched by newer consistent information by passing through f_{BC}, B and finally f_{AB}. In other words, expanding Equation 5:8 is resulting with the right combinations for computing exact inference because messages consistently propagated through the graph structure.

Criterion 3: No "double counting" [Pearl 1988] is allowed. Criterion 1 is a necessary but not sufficient condition to perform exact inference. In fact, in cyclic graphs nodes can receive the same message multiple times. Exact inference requires reconstructing the correct joint probability distribution table and "double counting" translates into errors.

Criterion 4: The imposition of carefully-selected initial conditions is necessary to block oscillations and to reach convergence. Oscillations arise when in some particularly regular cyclic graph Criterion 3 is not respected and each node periodically receives the same messages. With a proper choice of initial conditions, it is possible to schedule the messages arrival in order to avoid "double counting". In singly connected graphs, leaf nodes are always sending the same message by construction, and so convergence is guaranteed.

In the following Sections, we analyze Loopy BP performing approximate inference on cyclic graphs. Thus, we identify which one/s of the previous necessary criteria has/have been violated. Then we enforce all the criteria to fix the problem.

5.1.6 Single loop graph

Single loop graphs, in Figure 5:5 and Figure 5:6, are of particular theoretic interest because they are very similar to singly connected graphs but for a "closing" edge. However, this edge is responsible for the Criterion 4 to be violated. In fact, there are no leaf nodes that block oscillations. Indeed, Criterion 3 is violated, too.

Here we distinguish between single loop graphs according to N (the number of RVs they have), because this determines the opportune initial conditions required to enforce Criterion 4 and Criterion 3.

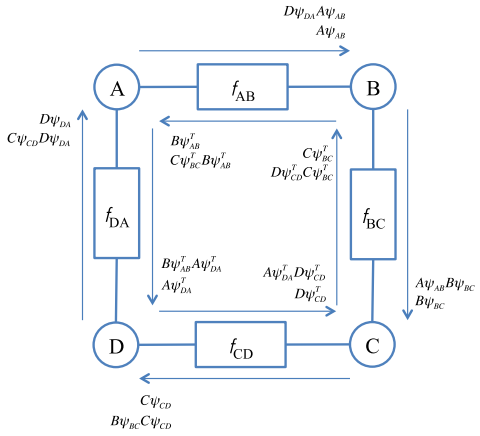
5.1.6.1 Even number of RVs

Consider the FG in Figure 5:5a with N = 4. After just two iterations, both Criterion 3 and Criterion 4 are violated as mentioned before. In fact, each node receives the message of its diagonal opposite twice. However, Criterion 1 is satisfied very quickly because of the

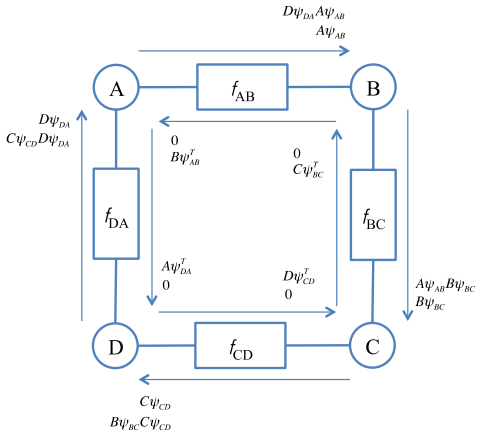
© 2017 Tous droits réservés.

lilliad.univ-lille.fr

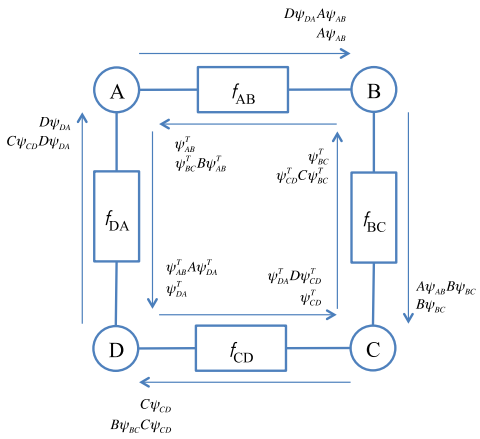
increased redundancy due to the loop. This observation is interesting, because it foments the intuition that if well managed, this redundancy can enhance the performance of BP to such an extent to prefer loopy graphs to singly connected ones, depending on N.



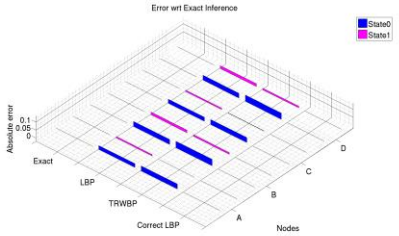
(a) Actual dynamics



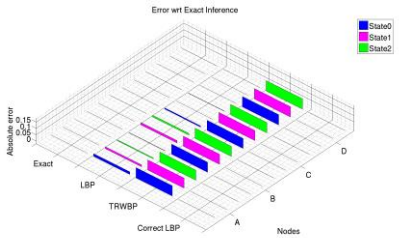
(b) Possible solution



(c) Final solution



(d) Simulation results. Binary RVs



(e) Simulation results. Ternary RVs

Figure 5:5. Exact inference on a single loop graph (N = 4).

A solution that satisfies Criterion 3 and Criterion 4, could be to choose as initial condition for each node, only one of the two edges to send their prior on the belief. However, as shown in Figure 5:5b, Criterion 1 is not satisfied with the convention chosen for iterations. In fact, every node is missing the message sent by one of the nonadjacent edges. Furthermore, if a third iteration is performed, again Criterion 3 will be violated.

Figure 5:5c shows the chosen solution that satisfies all the criteria. The initial condition for each RV node is to send its belief to one edge and a unitary message, i.e. the identity matrix, to the other. In this way, the factor's message is taken into account and propagates in the graph.

Figure 5:5d reports the simulation results where Loopy BP (LBP), Tree-ReParametrization BP (TRWBP) and our approach referred as Correct LBP are compared to the exact inference performed by exhaustive enumeration. In particular, the absolute error is computed

© 2017 Tous droits réservés.

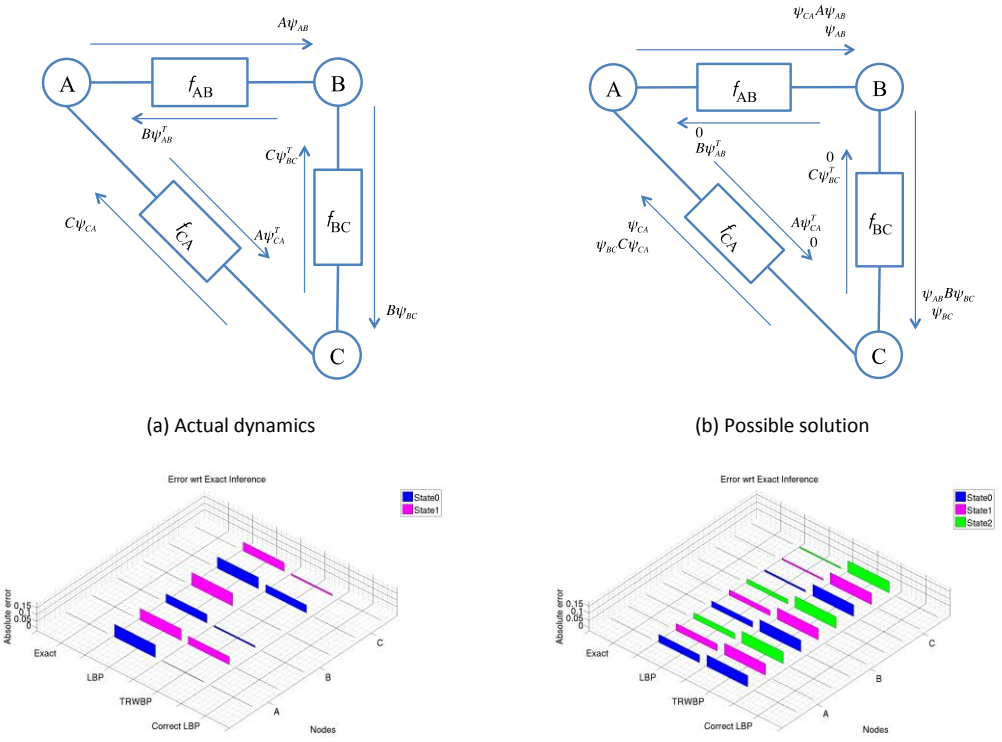
lilliad.univ-lille.fr

and shown. Our approach performs exact inference. This holds even for ternary RVs, Figure 5:5e. In this case, however, TRWBP does not converge and so it is assigned with an arbitrary large error value for a better visualization.

This solution turns out to be valid for general single loop graphs with even N. Convergence is, then, reached in N/2 iterations.

5.1.6.2 Odd number of RVs

Consider the FG in Figure 5:6a. After the first iteration, Criterion 1 is not satisfied because each RVs node is missing the message of the nonadjacent edge. However, further iterations will violate Criterion 3.



(a) Actual dynamics

(b) Possible solution

(c) Binary RVs simulations

(d) Ternary RVs simulations

Figure 5:6. Exact inference on a single loop graph (N = 3).

Therefore, as in the previous case, some specific initial conditions have to be adopted. In fact, Figure 5:6b shows a solution that satisfies all the criteria for exact inference, but requires an extra iteration. In practice, at the first iteration each node sends the identity message to one of its edges, while transmits nothing on the other.

Figure 5:6c reports the simulation results where our approach performed exact inference. As before, this remains valid also for ternary RVs, Figure 5:6d.

This solution is valid for general single loop graphs with odd N. Convergence is again reached in ⌈N/2⌉ iterations.

5.1.7 Toy turbo codes

In [Weiss 1997], a toy Turbo Codes structure was analyzed as an example of “balanced” network, i.e. a quasi-regular graph where oscillations are periodic due to an “equal double counting” for all nodes.

We perform our analysis on messages dynamics in the matrix formulation, Figure 5:7.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

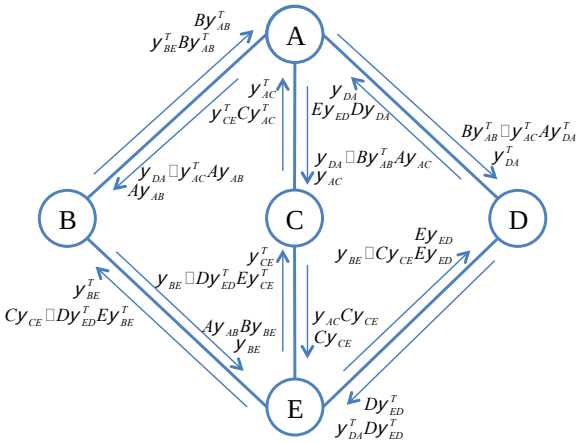


Figure 5:7. Toy Turbo Codes example.

The same initial conditions adopted for single loop graphs with even N result sufficient to satisfy all the criteria for exact inference, but Criterion 2.

In fact, in this example nodes A and E consist of three edges and so they are responsible for consistently combining incoming messages to produce the outgoing message without any loss of information. As shown in Figure 5:7, this has been done by applying the same mechanism exploited in Equation 5:10 for marginals computation, i.e. the element-wise product. However, for the message propagation task, this approach does not work because intermediate combinations of the incoming messages are suppressed. Thus, there is some information loss that is violating Criterion 1, as Criterion 2 is not satisfied.

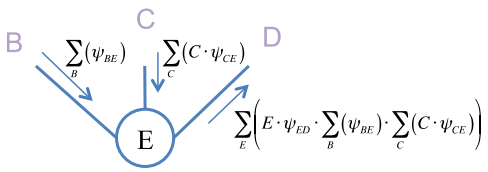
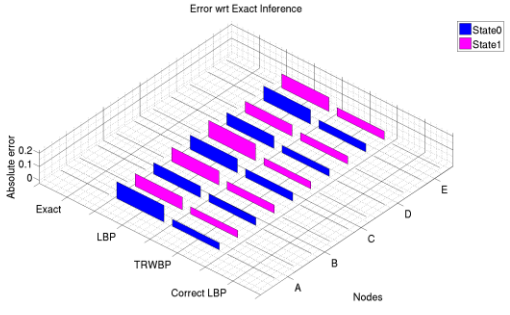


Figure 5:8. Consistent messages propagation on node E.

In order to better understand the problem, Figure 5:8 shows how the message from E to D is consistently computed by conventional message passing formulas, i.e. Equation 5:5 and Equation 5:6 for marginals computations. In this case, incoming messages from nodes B and C are consistently combined and no violations for the Criterion 2 are encountered. In fact, by expanding the resulting formula for the output message, all possible combinations are taken into consideration. In particular, intermediate combinations like $\psi_{BE}(B = b_0, E) \cdot [P(C = c_1)\psi_{CE}(C = c_1, E)]$ or $\psi_{BE}(B = b_1, E) \cdot [P(C = c_0)\psi_{CE}(C = c_0, E)]$ are not suppressed, resulting with exact inference.

To overcome this problem, and so, to make matrix formulation consistent according to Criterion 2, the combination of incoming messages is explicitly enforced. In particular, incoming messages are replicated twice and are sorted according the desired combinations to be obtained. At this point, the element-wise product is consistently performed. Of course, this requires the number of rows of the outputting message matrix to be doubled, but it does not compromise any further computation.



© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Matteo Causo, Lille 1, 2017

Figure 5:9. Simulation results for the toy Turbo Codes.

Simulation results are shown in Figure 5:9, where our approach performed exact inference.

5.1.8 Generalization

In our current work, we are applying the findings we have derived so far to arbitrary cyclic graphs. In fact, whether the criteria for exact inference apply in the generality of cases, how to select the initial conditions that satisfy those criteria still is an open question. Thus, this Section is intended to provide a practical and easy methodology to solve this problem after some general considerations on convergence and management of redundancy.

5.1.8.1 Management of redundancy

According to the redundancy considerations, arbitrary cyclic graphs can be classified as being somewhere between single loop graphs and fully connected ones. Furthermore, arbitrary cyclic graphs could present some singly connected paths. This categorization allows us to define some bounds for the convergence of Loopy BP by performing exact inference.

	# of nodes	# of edges	# of iter. for convergence
Singly connected	N	$N-1$	Diameter of the graph
Single loop	N	N	$\lceil N/2 \rceil$
Fully connected	N	$\sum_{i=1}^{N-1} i$	2

Table 5:2 Comparison among three main graph typologies.

Singly connected graphs are acyclic and are guaranteed to lead to exact inference, because they do not have to face redundancy. Their convergence is achieved in a number of iterations equal to the diameter of the graph [Mackay 2003]. In single loop graphs by managing the redundancy, convergence is achieved in $\lceil N/2 \rceil$ iterations, with N nodes. In fully connected graphs every node is directly connected with all the others. This represents the highest redundancy and set a lower bound for the minimum number of iterations needed to achieve convergence under opportune initial conditions, that is always 2. Table 5:2 summarizes this analysis and validates our thesis, i.e. redundancy actually boosts convergence.

5.1.8.2 Tree-ReParameterization

Tree-ReParametrization (TRW) proposed in [Wainwright et al. 2003], defined a valid technique for improving convergence while performing approximate inference in arbitrary cyclic graphs. Further improvements related to the sequentiality [Kolmogorov 2006] rather than the particularity [Meltzer et al. 2009] of the scheduling for the tree updates, could be shown to be derived from the satisfaction of the criteria stated in the present work.

Anyway, TRW could not be used to perform exact inference over arbitrary cyclic graphs. In fact, as argued in [Wainwright et al. 2003], “if the graph is cycle free, then there exists a unique reparameterization specified by exact marginal distributions over cliques. For a graph with cycles, on the other hand, exact factorizations exposing these marginals do not generally exist.” In other words, by enforcing Criterion 1 to be valid, each node of the arbitrary cyclic graph has to have a complete global vision of the whole graph. This is not the case if the exact inference is performed over a number of spanning acyclic graphs with limited vision. Further- more, Criterion 3 imposes not to have “double counting”. Thus, for each node of the arbitrary cyclic graph the resulting inference value has to be computed by a convenient combination of the exact inference values obtained from the spanning acyclic graphs involving that node.

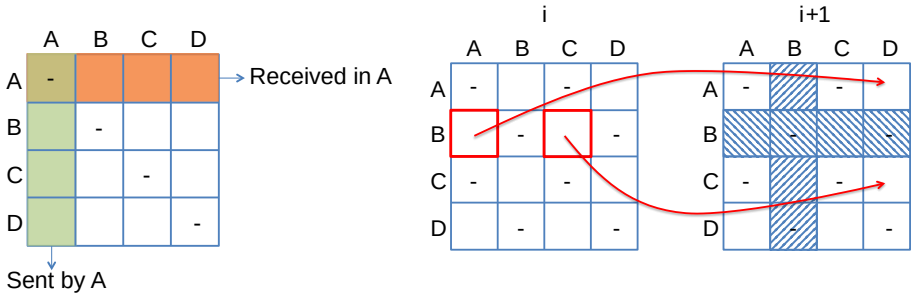
5.1.8.3 Consistent initial conditions

The single loop graphs and the toy Turbo Codes examples previously analyzed, successfully presented a practical way of performing exact inference over particularly regular cyclic graphs. In other words, it is sufficient to define the opportune initial conditions for the message passing together with the right number of iterations. This applies in general, but for particularly complex topologies of graphs some complementary ad-hoc solutions may have to be designed. For example, in [Weiss 1997] a graph consisting of both a single loop and a tree has been studied. In this case, exact inference could be firstly performed over the tree, so that after convergence, the tree collapses into the node connecting it to the single loop. Then, exact inference is performed on the single loop by exploiting the initial conditions previously presented. The reverse is also required in order to ensure all nodes to converge to exact inference. By enforcing all the criteria proposed in the present work, exact inference is guaranteed to be computed for the overall starting cyclic graph.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

An automatic methodology to define the opportune initial conditions and to discern between regular or ad-hoc solutions is therefore necessary. At this regard, we reassess the adjacency matrix[15] by providing it with new interpretations for rows and columns. In particular, for each RV node, its relative row accounts for all its incoming messages while its column for all its outgoing messages, Figure 5:10a. Then, we employ successive instances of this revisited adjacency matrix as an inversely evolving map of the message passing in the whole graph. Figure 5:10b shows the working principles together with the populating rules for the next iteration instance. In particular, the position BA of the instance at iteration i contains the outgoing message of the node A received by B. In order to track back this message we need to deprive it from the information added by A and to identify the node who sent this message to A. Therefore, this message has to be opportunely placed in the row relative to A into the instance at iteration $i+1$, in a column not related to B.

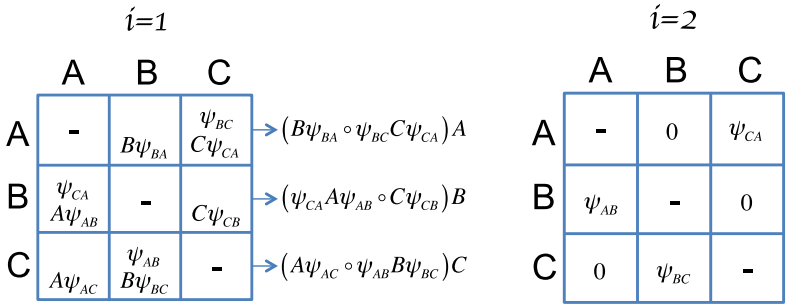


(a) Revisited instance (b) Populating rules

Figure 5:10. Inversely evolving map for message passing.

As a practical example, we derive the opportune initial conditions used in the single loop graph with $N = 3$ case.

Figure 5:11a shows the instance at iteration $i = 1$ that directly covers the joint probability distribution table. In fact, the product of all the elements for each row is proportional to $\tilde{P}(A, B, C)$. This represents the convergence state.



(a) Convergence state (b) Starting state

Figure 5:11. Initial conditions derivation example.

The instance at iteration $i = 2$ is populated by applying the rules previously exposed. This instance turns out to contain all elementary messages such as "nothing sent" 0; the identity message ψ_{XY} or the complete message $X\psi_{XY}$. Therefore the starting state is achieved and so are the initial conditions, i.e. each node sends the identity message to one of its edges, while transmits nothing on the other.

5.1.8.4 *Limits and observations*

The approach so far proposed allows to consistently detect the opportune initial conditions for arbitrary cyclic graphs. The starting state is reached in a number of iterations equal to the one required for the Loopy BP to converge by performing exact inference.

[15] An adjacency matrix is an $N \times N$ matrix that tracks the entirety of connections in a graph with N nodes. In particular if a node X is bidirectionally connected to a node Y, a "1" is present in both X Y and Y X matrix position.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Matteo Causo, Lille 1, 2017

Furthermore, elements in the starting state instance could be conveniently swapped with respect to the diagonal for providing each node with the same common rules.

Nevertheless, some collisions or contradictions could appear when populating the next iteration instance. One reason is related to the instance at the convergence state, i.e. iteration $i = 1$. Thus, it has to be reformulated by redistributing the messages from the joint probability distribution table in a different way. For particularly regular graphs, some rules could be derived for automatizing this procedure and preventing collisions. Anyhow, exhaustive tries until no collisions are accounted, is commonly valid.

If collisions could not be eradicated, the graph structure is particularly complex and so some ad-hoc solutions have to be figured out in order to satisfy all the criteria for exact inference.

5.1.9 Discussion

We analyzed BP performing exact inference over singly connected graphs and we extracted some generally valid criteria to allow Loopy BP to perform exact inference over arbitrary cyclic graphs. In order to enforce those criteria to be satisfied, opportune initial conditions have to be respected for message passing. A practical tool to choose these initial conditions is proposed by reconsidering the adjacency matrix as an inversely evolving map of the message passing, going from the convergence state back to the starting one.

Therefore exploiting these findings for real-world applications is expected to positively influence the popularity of BP algorithms. Furthermore, by performing exact inference over arbitrary cyclic graphs, convergence is guaranteed and so, Loopy BP would be chosen over Sequential Tree-ReParametrization algorithm for particularly time-demanding applications. In fact, as previously discussed the management of redundancy is actually boosting convergence, as shown in Table 5:3 where the number of iterations for Loopy BP, Tree-ReParametrization and Correct Loopy BP are compared for each example accounted in the present work. This is of paramount importance for practical hardware implementations, where a relaxed timing constraint could be translated into enhanced performances or improved energy-efficiency trade-off.

	Loopy BP	TRW	Correct Loopy BP
Singly connected (N=3)	2	2	2
Single loop (N=4)	13	16	2
Single loop (N=3)	10	20	2
Toy turbo codes	16	21	2

Table 5:3 Number of iterations for achieving convergence.

Furthermore, in [Farinelli et al. 2008] the generic problem of maximizing social welfare within a group of interacting agents in a MAS with decentralized coordination, is solved by message passing over a cyclic bipartite factor graph. Therefore our findings can be practically applied to improve convergence by performing exact inference. However, the paramount interest in decentralizing the coordination task is to make the system more reliable to failures and to allow it to dynamically scale as the number of agent varies. This implies, a dynamical definition of the opportune initial conditions, which highlights the limitations of the proposed approach by inversely evolving map. Therefore, future work could focus on improving this tool in order to prevent collisions in particularly regular cyclic graph topologies and to provide a better understanding of when and how to apply ad-hoc solutions.

5.2 Improving EM through inference

The Expectation-Maximization (EM) algorithm, introduced in Section 2.1.2, consists of two steps. The M-step computes the Maximum Likelihood Estimate (MLE) of the model parameters, while the E-step defines the degree of correlation between the data and the model behind the learnt parameters. The EM is proven to always increase the learning accuracy [Barber 2011], but it is guaranteed to converge to a local maximum of the likelihood, not to the global one [Rabiner 1989].

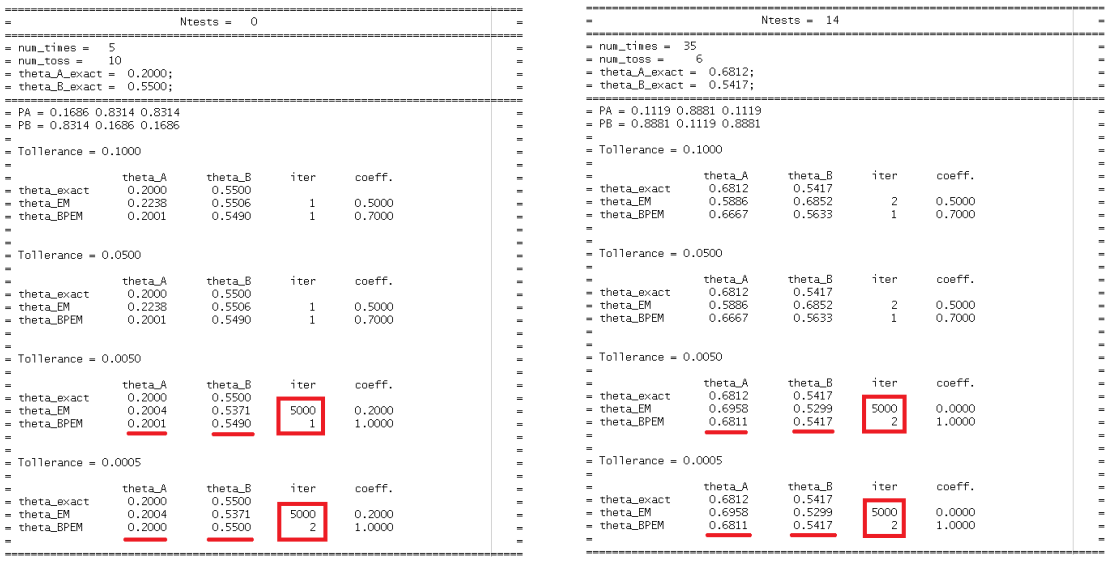
In [Roche 2011], several EM variants are presented to solve standard EM convergence problems. They are grouped in two categories, namely deterministic and stochastic EM variants. Deterministic variants attempt at speeding up the EM algorithm, either by simplifying computations or by directly increasing the convergence rate. Conversely, stochastic variants are more concerned with other limitations such as computing an integral that may not be tractable in some situations. By replacing tedious computations with stochastic simulations, the modified algorithm presents a lesser tendency to getting trapped in local maxima, yielding enhanced global convergence properties.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Our approach to improve the EM convergence and overall performance consists in providing the E-step with some mechanisms of inference propagation in order to get information from the whole dataset in a balanced manner. In fact, with such a comprehensive view, we aim at escaping local solutions and reaching more quickly and directly the right global one. Furthermore, distributed datasets can be locally elaborated and at the same time contribute to the system-level global solution, therefore allowing for a spatially distributed EM algorithm.

For the coin-tossing experiment in Section 2.1, the E-step evaluates $p(\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{\theta})$ and completes the information set with the identity of the coin that generated the observations. This probability can be computed through exact inference on the Bayesian Network in Figure 2:1 through Belief Propagation (BP) mechanisms because of its tree shape. We refer to this approach as Belief Propagation Expectation-Maximization (BPEM).



```
Ntests =  0
num_times =  5
num_toss =  10
theta_A_exact =  0.2000;
theta_B_exact =  0.5500;
PA = 0.1686 0.8314 0.8314
PB = 0.8314 0.1686 0.1686
Tollerance = 0.1000
                theta_A     theta_B     iter     coeff.
theta_exact      0.2000      0.5500
theta_EM         0.2238      0.5506        1     0.5000
theta_BPEM       0.2001      0.5490        1     0.7000
Tollerance = 0.0500
                theta_A     theta_B     iter     coeff.
theta_exact      0.2000      0.5500
theta_EM         0.2238      0.5506        1     0.5000
theta_BPEM       0.2001      0.5490        1     0.7000
Tollerance = 0.0050
                theta_A     theta_B     iter     coeff.
theta_exact      0.2000      0.5500
theta_EM         0.2004      0.5371     5000     0.2000
theta_BPEM       0.2001      0.5490        1     1.0000
Tollerance = 0.0005
                theta_A     theta_B     iter     coeff.
theta_exact      0.2000      0.5500
theta_EM         0.2004      0.5371     5000     0.2000
theta_BPEM       0.2000      0.5500        2     1.0000
```

```
Ntests =  14
num_times =  35
num_toss =  6
theta_A_exact =  0.6812;
theta_B_exact =  0.5417;
PA = 0.1119 0.8881 0.1119
PB = 0.8881 0.1119 0.8881
Tollerance = 0.1000
                theta_A     theta_B     iter     coeff.
theta_exact      0.6812      0.5417
theta_EM         0.5886      0.6852        2     0.5000
theta_BPEM       0.6667      0.5633        1     0.7000
Tollerance = 0.0500
                theta_A     theta_B     iter     coeff.
theta_exact      0.6812      0.5417
theta_EM         0.5886      0.6852        2     0.5000
theta_BPEM       0.6667      0.5633        1     0.7000
Tollerance = 0.0050
                theta_A     theta_B     iter     coeff.
theta_exact      0.6812      0.5417
theta_EM         0.6958      0.5299     5000     0.0000
theta_BPEM       0.6811      0.5417        2     1.0000
Tollerance = 0.0005
                theta_A     theta_B     iter     coeff.
theta_exact      0.6812      0.5417
theta_EM         0.6958      0.5299     5000     0.0000
theta_BPEM       0.6811      0.5417        2     1.0000
```

Figure 5:12. Comparison between EM and BPEM on the coin-tossing experiment.

We compared the EM and the BPEM on the coin-tossing experiment for estimating the biases $\boldsymbol{\theta} = \{\theta_A = 0.22; \theta_B = 0.55\}$ of two unfair coins, A and B, by analyzing 5 sets of 10 tosses each. Each set generated by a coin randomly chosen.

Results are stunning as shown in Figure 5:12: the BPEM improves the EM convergence and always results with more accurate estimates for the model parameters $\boldsymbol{\theta}$ than the standard EM does. Successively, we exacerbated the stop-learning criteria by choosing an increasingly small tolerance. In particular, both algorithms are iterated until the learnt parameters deviate for less than a permitted tolerance from exact values. Consequently, the number of iterations required for the EM algorithm to converge exploded at the maximum allowed, i.e. 5000, while the BPEM generally reached convergence after one or two iterations. We repeated the overall test several times on randomly defined datasets. In fact, in Figure 5:12Right the same astonishing results were found for a dataset consisting of 35 sets of 6 tosses each and model parameters $\boldsymbol{\theta} = \{\theta_A = 0.6812; \theta_B = 0.5417\}$.

5.2.1 Test benches

Test bench in Figure 5:13 compares the EM and the BPEM by evaluating their performance with a mark, according to the criteria summarized in Table 5:4 and Table 5:5. It automatizes the experimental setup definition to perform a big amount of tests and to come out with some statistics on the overall convergence.

Coefficient for better performances	
1.00	Both thetas are in the tolerance and both are more accurate than those of the other approach. Also the # of iterations is smaller.
0.85	Both thetas are in the tolerance and both are more than or as accurate as those of the other approach. But the # of iterations is not smaller.
0.70	Both thetas are in the tolerance and the # of iterations is smaller, but either only one theta or none is more accurate than that of the other approach.
0.50	Both thetas are in the tolerance, but the # of iterations is not smaller and either only one theta or none is more accurate than those of the other approach.
0.30	Only one theta is in the tolerance and it is more accurate than that of the other approach.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

0.20	Only one theta is in the tolerance.
0.00	No theta is in the demanded tolerance.
0.00	At least a theta is NaN and results diverged.

Table 5:4 Summary for the coefficient for better performances.

The coefficient reported in Table 5:4 defines which one of the two approaches performs better and qualifies this “better”. On the contrary, the coefficient in Table 5:5 relates the results to the permitted tolerance, thus it quantifies the “difficulty” of the test.

Coefficient per tolerance bracket	
0.7	Tolerance = 0.1
1.0	Tolerance = 0.05
1.5	Tolerance = 0.005
2.0	Tolerance = 0.0005

Table 5:5 Summary for the coefficient per tolerance bracket.

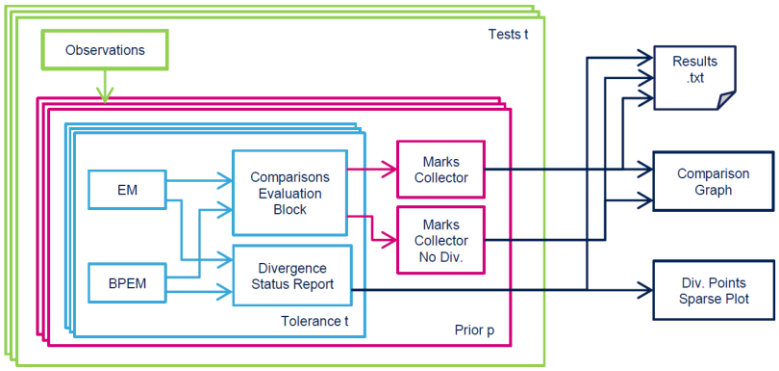


Figure 5:13. Block diagram of the test bench.

(EM and BPEM are exhaustively compared on several tests. For each test, new model parameters are chosen and a new dataset is randomly generated accordingly. Furthermore, for each test several priors are employed to study the convergence property in function of initial conditions. For each prior, the two approaches are compared for each tolerance bracket listed in Table 5:5. Comparisons results are elaborated by the Comparisons Evaluation Block that marks each approach exclusively on converged solutions or on overall results. For each prior, marks are then collected in the log-file “Results.txt” and plotted on a graph. The Divergence Status Report takes trace of not converged solutions and records their possible causes.)

The outcome of the test bench in Figure 5:13 is a log file, similar to those in Figure 5:12 and some statistics on the results, shown in Figure 5:14. The highest mark that could be obtained in total is 5.2, that is the sum of all the highest marks computed per each tolerance bracket by multiplying the value of the "coefficient for better performances" in Table 5:4, times the respective "coefficient per tolerance bracket" in Table 5:5. In Figure 5:14, normalized marks are shown for the ease of comparison.

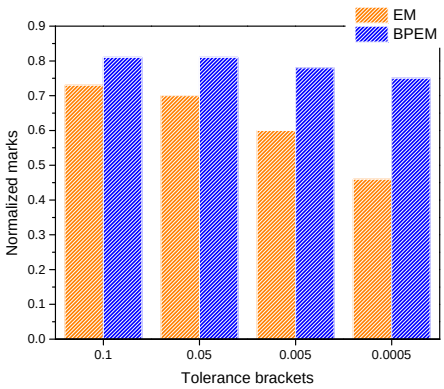


Figure 5:14. Outcome of test bench 2.0: EM vs BPEM.

Figure 5:14, together with the reports in Figure 5:12, clearly shows the effects of our optimizations to the EM algorithm. In particular, in Figure 5:14, the BPEM exceeds the EM performance for each tolerance bracket up to a 40% of improvement in the whole parameter space. Nevertheless, these results are due to the fact the EM has a tendency to converge to a finite value, but incorrect or roughly

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Matteo Causo, Lille 1, 2017

approximated. In fact, taking a look at the detailed reports, it is not uncommon that the EM will not terminate before the end of the maximum number of iterations, i.e. 5000, because it does not converge to the correct values. Conversely, the BPEM increases the convergence rate and the convergence towards the correct values.

5.2.2 MAPEM

Another very popular form of inference on PGM is the Loopy BP explored in Section 5.1. By the way, the problem of the coin-tossing experiment relies on a tree-shaped PGM and so the Loopy BP results into standard BP.

The Maximum A Posteriori (MAP) estimate can however substitute the BP. The MAP is closely related to Maximum Likelihood Estimate (MLE), but it employs an augmented optimization objective that incorporates a prior distribution over the quantity to estimate, as explained in Section 2.1. MAP estimation can therefore be seen as a regularization of MLE, because it avoids its over-fitting problems.

For what concerns inference, the MAP may result with a not unique solution, but it can be applied in more general cases and is more widely supported by the theory than BP, which is however "less fragile"; more confident in its solutions and more suitable for decision-making support.

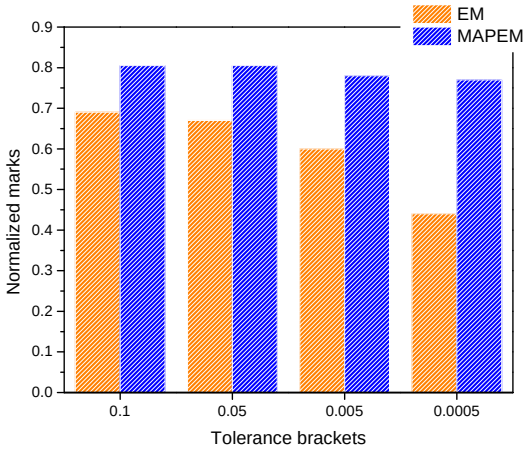


Figure 5:15. Outcome of test bench: EM vs MAPEM.

We derive and test the Maximum A Posteriori Expectation-Maximization (MAPEM). PGM theory ensures that is possible to derive the MAP formulation on the BN of Figure 2:1. The only difference with respect to BP is that summations are not used anymore, but are replaced with the maximum conditional probability value. Furthermore, the joint distribution is computed into the log-domain. In this way, products become summations and calculations results easier. The ease in deriving the MAPEM approach makes of it a possible alternative to BPEM for a simpler HW implementation.

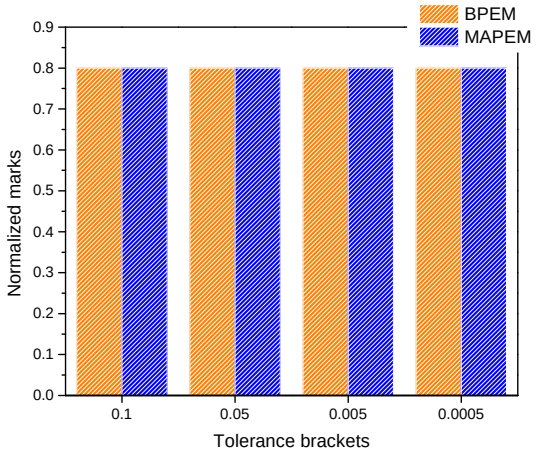


Figure 5:16. Outcome of test bench 2.0: BPEM vs MAPEM.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Matteo Causo, Lille 1, 2017

The same test bench as in the previous Subsection, has been used for evaluating the performances of MAPEM with respect to both EM and BPEM as respectively shown in Figure 5:15 and Figure 5:16. Results are again surprisingly excellent. The MAPEM performs better than the EM in terms of final results and convergence rate as it was for the BPEM. However, in comparison with this latter, both accuracy and convergence are exactly the same. This is due to the fact that we are performing exact inference on the same tree-shaped graph for both approaches and therefore we get the same results. This can be seen as a counter-check of validity of our analysis.

5.3 Sampling Modulation

The advancement in low-power embedded technologies is having a pronounced impact on the fields of medical diagnosis and rehabilitation. For the emerging wearable and implantable medical systems to be of practical value however, it is essential that information can be reliably extracted over extended periods of time [Ramgopal et al. 2014]. Body biopotentials are among the most valued biomarkers exploited in today's wearable devices. To turn the raw data into useful information, many feature extraction techniques are applied both in time and frequency domain, ranging from simple filtering to complex transforms [Tzallas et al. 2009]. One of the prominent examples of such, are feature extraction techniques used on Electroencephalogram (EEG) signals [Sanei and Chambers 2007]. EEG results from the non-stationary processes that occur during neurons activation, and is composed from the superposition of local current flows produced during synaptic excitations. The observable frequency spectrum stays within five frequency bands named delta (0.5 - 4 Hz), theta (4-8 Hz), alpha (8-13 Hz), beta (13-22 Hz) and gamma (30-40 Hz). In these, the detection of amplitude and frequency changes is of prime interest, especially to recognize the occurrence of certain phenomena, such as seizure. The simple calculation of the signal energy is not sufficient to detect variation in frequency and amplitude, as it superimposes information from all bands. Hence, the most used techniques are the frequency and time-frequency analyses, with the Discrete Wavelet Transform (DWT) being one of the most effective and representing the state-of-the-art solution.

The calculation of energy of the DWT Detail Coefficients gives quantitative information on the frequency content of the EEG sub-bands. This information can then be used, for instance, to train a supervised classification algorithm, such as Support Vector Machine (SVM) or Artificial Neural Network (ANN) that deals with the variability and the high noise contents of the incoming signals. These approaches and their implementations have been demonstrated to yield good performance in terms of recognition accuracy [Panda et al. 2010; Nunes et al. 2014]. Nevertheless, the DWT is inherently complex, and its computational cost puts a burden on the energy-starved wearable sensor systems.

We propose a substantially simpler time-frequency information extraction technique for biosignal processing called Sampling Modulation (SM). It extracts information on variations of amplitude and frequency of the EEG signal in the time domain using a footprint of finite differences after signal windowing. The algorithm is verified by comparing it with DWT in a typical seizure detection application [Liu et al. 2012]. The two approaches are compared with respect to accuracy, sensitivity, specificity, energy efficiency and performance after implementation on FPGA platform.

5.3.1 Feature extraction techniques

The analysis of biomedical signals such as ExG, i.e. EEG, Electromyography (EMG) and Electrocardiogram (ECG), requires pre-processing to detect and evaluate an event occurrence in time, triggered by a signal change in frequency, amplitude and/or phase.

Therefore, there has been a growing interest in time-frequency decomposition tools like the DWT. The Wavelet Transform provides a time-frequency representation of a non-stationary signal, giving information on the frequency content of a signal in the time domain. This is done through a filter bank that processes the input signal by decomposing it in high and low frequency components. For a DWT decomposition of level n, the coefficients of the LPFs are named Detail Coefficients (D_n) while the coefficients of the HPFs are named Approximation Coefficients (A_n). The Detail Coefficients of the DWT decomposition return the information on the energy content of the given signal.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

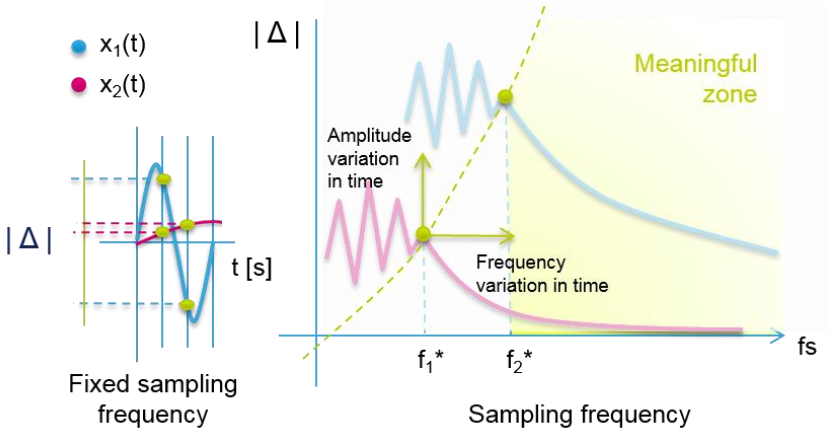


Figure 5:17. Sampling Modulation working principle.

(Two time-invariant signals $x_1(t)$ and $x_2(t)$ are sub-sampled with a sampling frequency $f_{s_{Nyq}} < f_s < \hat{f}_s$. In the image on the left, the basic principle behind SM is illustrated. The difference of two consecutive samples is computed for each f_s. In the image on the right, the resulting representation (unique signature) in the frequency domain for the two signals that are easily separable in the "Meaningful zone" where they show a monotonically decreasing behavior. Finally, by assuming $x_2(t)$ to be time-variant, both possible amplitude and frequency variations in time are indicated through the green arrows.)

The idea behind Sampling Modulation (SM) is to discern and highlight all kinds of time-variant changes in the analyzed signals by modulating the sampling frequency. In particular, SM provides the signal with a unique signature (representation) in the frequency domain, i.e. the signal amplitude, frequency and phase characteristics are summarizes in a single feature that preserves time-variant information. To illustrate the concept, let us first consider two time-invariant signals $x_1(t)$ and $x_2(t)$ with different amplitude $A_{x_1} \neq A_{x_2}$ and frequency $f_{x_1} \neq f_{x_2}$ that are sampled at the same sampling frequency $\hat{f}_s \gg f_{s_{Nyq}}$, where $f_{s_{Nyq}} = 2\max\{f_{x_1}, f_{x_2}\}$, so as to satisfy the Nyquist-Shannon sampling theorem. SM is able to decisively separate the two signals, as shown in Figure 5:17.

The basic principle of SM is to subsample the signal by modulating the sampling frequency f_s in the $\left[f_{s_{Nyq}}, \hat{f}_s\right]$ range and to compute the difference, in absolute value, of two consecutive samples. To discern between the two signals, it is essential that the analysis is performed in the zone referred to as "Meaningful zone" in Figure 5:17. In fact, for sampling frequencies close to $f_{s_{Nyq}}$, consecutive samples are generally too different to show a coherent meaningful behavior, while for high sampling frequencies, they are too close to show significant differences. Therefore, there will always be a sampling frequency f_x^* such that for higher sampling frequencies the difference between two consecutive samples assumes, in absolute value, a decreasing monotonic behavior.

Mathematically, the SM could be studied with the finite differences forward formula Equation 5:12, where $x(t)$ is the signal and f_s the sampling frequency.

$$\left|\Delta_{1/f_s}\right|[x](t) = |x(t + 1/f_s) - x(t)|$$

Equation 5:12 – Finite differences forward formula.

$$\lim_{f_s \to \infty} \left|\Delta_{1/f_s}\right|[x](t) = \lim_{1/f_s \to 0} |x(t + 1/f_s) - x(t)| = 0$$

Equation 5:13 – Proof of asymptotic convergence to zero.

The asymptotic convergence to zero is trivially proved by the limit in Equation 5:13, while the extrema for $x(t)$ are computed with Equation 5:14, where f is the signal frequency; ϕ the signal phase and $n \in \mathbb{N}$ allows for choosing the last extremum such that a monotonic convergence to zero is ensured.

$$f_x^* = \frac{2\pi f}{[(2n+1)\pi]/2 - 2\pi f t - \phi}$$

Equation 5:14 – Extrema for $x(t)$.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Matteo Causo, Lille 1, 2017

By relaxing the time-invariance assumption, time-variant changes in the analyzed signal amplitude and frequency be-come independent and easily recognizable, as they results respectively in a vertical and a horizontal translation, as shown for the signal $x_2(t)$ in Figure 5:17.

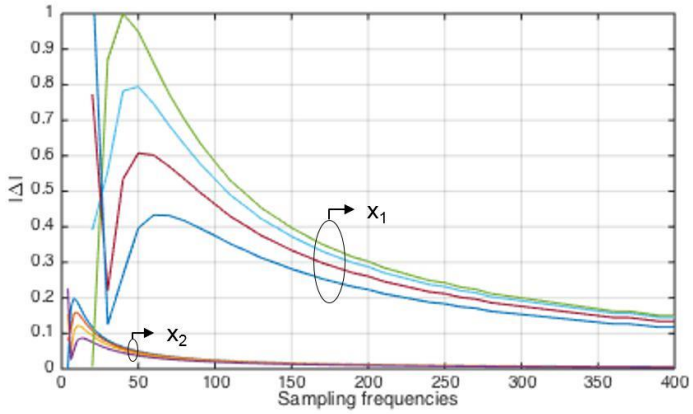


Figure 5:18 Time-variant signals analysis.

(MatLab simulation for the SM applied on two time-variant signals $x_1(t)$ and $x_2(t)$. Only phase variations are allowed in time, but they can be confused with both small amplitude and frequency variations because they involve signal translation in both vertical and horizontal directions. In fact, a family of curves is generated around the time-invariant curves plotted in Fig. 2. Anyhow the two signals are still easily separable, especially in the ”Meaningful zone”.)

Time-variant phase changes however affect both directions and can be confused with both amplitude and frequency variation, as it is shown in Figure 5:18. Horizontal variations are explained with Equation 5:14 through the relations between phase and frequency, while vertical variations follow from Equation 5:12. Nevertheless, if we take into account the time-variant phase error according to $\phi(t) = \phi + \delta_\phi(f_s)$ then Equation 5:14 becomes Equation 5:15, where the error $\delta_\phi(f_s)$ is shown to be practically absorbed when computing the approximation error for the frequencies of interest, i.e. in the “Meaningful zone”.

$$f_x^* = \frac{2\pi f}{[(2n+1)\pi]/2 - 2\pi f t - \phi - \delta_\phi(f_x^*)}$$

Equation 5:15 – Extrema for $x(t)$ with rime-variant phase.

The two signals are however separated in the “Meaningful zone”, therefore a binary classifier is able to easily learn to discern between them. Therefore, SM acts as a feature extraction engine, not just as a preprocessing. In fact, it does not require further elaborations to extract energy from detailed coefficients like in the case of the DWT.

5.3.2 Algorithm

The implementation of SM results in a simple and intuitive algorithm. It derives directly from the application of Equation 5:12 on consecutive windows of the analyzed signal, as shown in Algorithm 3. Therefore, time-variance is preserved like for a time-frequency transform, e.g. the DWT.

ALGORITHM 3. Sampling Modulation (SM)

load buffer b
compute positions in $posList$
for all w **in** windows **do**
  for all p_{f_s} **in** $posList$ **do**
    $|\Delta_{w,f_s}| = |b(1) - b(p_{f_s})|$
  end for
end for

Figure 5:19 shows the computing principle for the EEG signal example where the window width is of 256 samples, i.e. equivalent to a typical 1 second of recording.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

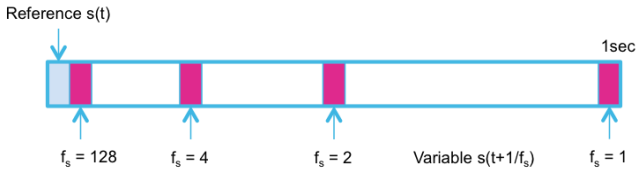


Figure 5:19 Schematic illustration of the basic principle behind Algorithm 1 implementing SM.

(For the EEG recording analysis, a buffer of 256 samples is loaded per each 1-second-width window of the EEG signal. SM is finally computed by performing successive subtractions between the reference sample in position 1 and consecutive samples at positions p_{f_s})

For each window, a buffer of samples is loaded and successive subtractions are performed between the reference sample and those identified by a position p_{f_s} relative to the modulated sampling frequencies. However, for a real-world application, signals are sampled to a fixed sampling frequency $\hat{f}_s$ and therefore we dispose of a finite number of samples in discrete positions that cannot be referred by the continuous modulation of the sampling frequency, because of the quantization error due to the approximation to integer positions in Equation 5:16.

$$p_{f_s} = \lfloor \hat{f}_s / f_s \rfloor$$

Equation 5:16 – Position relative to the modulated sampling frequency.

Indeed, for f_s in the range [103-170]Hz the sample $x(t + 1/128)$ will be chosen because $p_{[103;170]} = p_{128}$.

SM has the advantage of simplicity, compactness and a maximally parallel implementation. Thus, for each window the computing overhead is drastically reduced with respect to other transforms such as the DWT, as only a limited number of samples is taken into account.

5.3.3 Experimental results and FPGA implementation

SM is a powerful and very versatile tool for real-time feature extraction on time-variant signals such as the biomedical ones. We have tested its performance with respect to the DWT on EEG signals for seizure detection on the CHB-MIT database [Shoeb and Guttag 2010]. The Children's Hospital Boston - Massachusetts Institute of Technology (CHB-MIT) database consists of EEG recordings from 22 pediatric subjects with intractable seizures. All signals are sampled at 256Hz with 16-bit resolution according to the international 10-20 system of EEG electrode positions. Data are labeled in order to train a supervised Machine Learning (ML) classifier and to evaluate its outcomes during the testing phase.

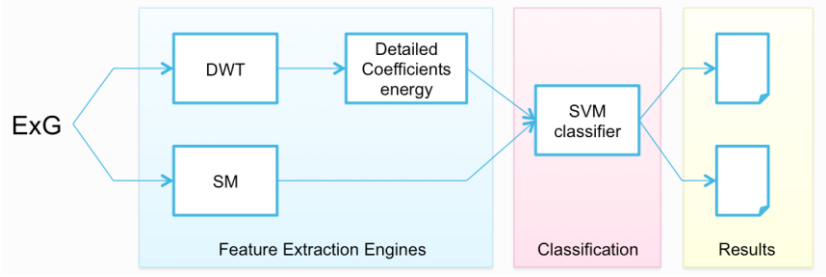


Figure 5:20 Software testing setup for comparing SM vs. DWT.

(A biopotential (ExG) is divided into consecutive windows and it is presented to both the feature extraction engines, which results feed the binary SVM classifier. Classification results are collected and statistics such as classification accuracy, sensitivity and specificity are computed.)

The testing setup for the seizure detection application is described in Figure 5:20. We have employed SM and DWT as feature extraction engines for the EEG biomedical recordings. Their outcome has served as input for the binary SVM classifier with Gaussian kernel. Statistics on classification performance have been finally collected and are presented in Table 5:6 in terms of average classification accuracy, sensitivity and specificity among the overall database. These results are in line with the state-of-the-art of seizure detection application [Nunes et al. 2014]. In this test the objective of the classification has been simplified to discern between two classes, i.e. either seizure or normal activity. Therefore, the final testing accuracy for both SM and DWT features is artificially high. However, sensitivity values are not as good as the specificity ones, meaning that the onset of seizure is sometimes missed, causing False Negatives to occur. The same behavior has been confirmed through ROC analysis.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Matteo Causo, Lille 1, 2017

	Accuracy	Sensitivity	Specificity
DWT	99.50 %	0.7883	0.9971
SM	99.68 %	0.8405	0.9978

Table 5:6 Final classification results for the testing setup in Figure 5:20.

The general result is that features extracted with SM are in average slightly better in overall classification performance than those extracted with DWT. Moreover, as shown in Figure 5:20, DWT requires a computational overhead to provide the energy of its Detailed Coefficients as a feature. In order to quantify this complexity overhead in terms of execution time, power and area consumption, we have synthesized and implemented both SM and DWT on a Xilinx Z-7020 FPGA platform.

Each EEG record is consequently stored in an external memory modeled as a single-port RAM. A Finite State Machine (FSM) controller is in charge of computing the reading/writing addresses from/to the memory and to orchestrate the whole data-flow in order to perform a correct signal windowing and processing. The Device Under Test (DUT) is either the SM or the DWT with the block for computing Detailed Coefficient energy, as shown in Figure 5:20. Outcomes are collected and presented to a MatLab script for the classification and the computation of final statistics.

	Area		Power	Execution Time	Energy/record
	LUT	FF	[mW] @ 25 MHz	[msec]	[µJ]
DWT	446	342	5	6.349	31.745
SM	136	107	1	0.154	0.154
Improvement SM wrt DWT	3x		5x	41x	206x

Table 5:7 FPGA implementation characteristics for both DWT and SM.

The implementation results are summarized in Table 5:7. As expected, SM requires less area than the DWT because of the further computational step for extracting DWT Detailed Coefficients energy. Therefore, the gain in area consumption is of 3x. The execution time per complete EEG record, i.e. 1 hour in average, is reduced as well, with a favorable factor of 41x. There are two main reasons behind this result. Firstly, as just stated, SM doesn't need further elaboration for feature extraction, while DWT does. Secondly, SM does not require all the samples to be elaborated, rather the few at the significant positions defined by Equation 5:16. Power consumption reduction is on the order of 5x and the estimated energy per record classification of SM is 154 nJ, i.e. 206x lower than DWT requires, thus making SM particularly suitable for applications with strict requirements on size, computational cost and power consumption such as implantable and wearable electronics.

5.4 Internships proposition and tutoring experience

During the present PhD thesis work, the author has been required to formulate and tutor three internships in the context of Machine Learning. In particular, the author was challenged in identifying three or more subjects related to the present PhD thesis work to be preparatory for the formation of three bachelor students and their introduction into the world of scientific research.

The proposed topics are detailed in the Subsections below.

5.4.1 Exact inference on arbitrary cyclic graphs

Message passing algorithms like Belief Propagation (BP) allow to perform a global task by elaborating local data in a distributed concurrent manner. They have been copiously employed for probabilistic inference over graphical models. However, they are guaranteed to result with exact inference only for singly connected graphs. For arbitrary cyclic graphs only approximate inference can be performed and the brute application of BP is referred as Loopy BP. Surprisingly, Loopy BP gives excellent results in practical applications, such as error correcting codes (Turbo Code), speech recognition, image understanding, stereo matching, photomontage, background segmentation, and de-noising in computer vision. However, Loopy BP is not guarantee to converge and a theoretical understanding of the algorithms behavior has yet to be achieved. Recently, the increasing interest in designing and building unconventional hardware has pushed the Electronics community to approach Artificial Intelligence and Statistics in order to exploit these techniques and tools. Therefore, a better understanding and control of the BP dynamics in an arbitrary network is of paramount interest for future innovative hardware implementations.

Following the preliminary studies in Section 5.1, the candidate was required to gain an overall and specific understanding of the subject; to report on existing solutions for exact inference and their limitations; to define a mathematical framework where to explain Loopy BP dynamics; to provide a network simulator tool (MatLab, C, Java, …) and to estimate qualitatively its complexity.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Matteo Causo, Lille 1, 2017

5.4.2 Learning from Deep Learning: extracting invariant features.

Deep Learning (DL) has allowed Artificial Neural Networks (ANNs) to succeed in many image recognition and classification competitions where predictions should be unchanged, or invariant, under one or more transformations of the input variables. ANNs can learn the invariance, at least approximately if sufficiently large numbers of training patterns are available. However, thanks to DL the invariance property has been built into the structure of the ANNs allowing for even better performance. The problem is that DL requires huge computational resources and a huge memory to store the system parameters. Furthermore, its hardware acceleration is limited by the communication bottleneck with the memory.

Invariance can also be built into a separate preprocessing engine by extracting features that are invariant under the required transformations, so that DL could be replaced by a lighter and simpler Machine Learning algorithm. Therefore invariance principles inspired to DL could lead external Feature Extraction Engines (FEE) to fit into portable and wearable electronics with limited resources constraints.

The candidate was required to gain an overall and specific understanding of the subject; report on existing solutions for feature extraction in cognitive applications; define invariance principles inspired to Deep Learning; provide a software-base prove-of-concept (MatLab, C, Java, …) and estimate qualitatively its complexity.

A summary of the internship work and its outcomes is in Appendix 7.1.

5.4.3 The speaker diarization problem

Speaker diarization is the problem of segmenting an audio recording of a meeting into temporal segments corresponding to individual speakers. So, it is a combination of speaker segmentation and speaker clustering. The problem is rendered particularly difficult because the number of people participating in the meeting is not known a priori. One of the most popular methods is to use a Gaussian Mixture Model (GMM) to model each of the speakers, and assign the corresponding frames for each speaker with the help of a Hidden Markov Model (HMM). Other approaches rely on Bayesian nonparametric solutions, like Hierarchical Dirichlet Process Hidden Markov Model (HDP-HMM). Although these are very powerful statistical tools they tend to over-segment the audio data by creating redundant states and rapidly switching among them. Therefore they result with poor classification accuracy.

The candidate was required to gain an overall and specific understanding of the subject; report on existing solutions for the speaker diarization problem; define the input stream partitioning criteria (Window size, sampling rate, preprocessing) to increase final classification accuracy; provide a software-base prove-of-concept (MatLab, C, Java, …) and estimate qualitatively its complexity.

A summary of the internship work and its outcomes is in Appendix 7.2.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Matteo Causo, Lille 1, 2017

References

Srinivas M. Aji and Robert J. Mceliece. 2000. The Generalized Distributive Law. 46, 2 (2000), 325–343.

David Barber. 2011. Bayesian Reasoning and Machine Learning. *Mach. Learn.* (2011), 646. DOI:http://dx.doi.org/10.1017/CBO9780511804779

Claude Berrou, Alain Glavieux, and Punya Thitimajshima. 1993. Near Shannon Limit Error - Correcting Coding and Decoding : Turbo-Codes (1). *Commun. 1993. ICC '93 Geneva. Tech. Program, Conf. Rec. IEEE Int. Conf.* , 1 (1993), 1064–1070 vol.2. DOI:http://dx.doi.org/10.1109/ICC.1993.397441

Gregory F. Cooper. 1990. The computational complexity of probabilistic inference using bayesian belief networks. *Artif. Intell.* 42, 2–3 (1990), 393–405. DOI:http://dx.doi.org/10.1016/0004-3702(90)90060-D

A. Farinelli, A. Rogers, A. Petcu, and N.R. Jennings. 2008. Decentralised Coordination of Low-Power Embedded Devices Using the Max-Sum Algorithm. , Aamas (2008), 12–16.

Zoubin Ghahramani. 2002. Graphical models : parameter learning. *Handb. Brain Theory Neural Networks* , 2 (2002), 486–490. DOI:http://dx.doi.org/10.1561/2200000001

Michael I. Jordan and Yair Weiss. 1996. Probabilistic inference in graphical models. *Lauritzen, S. L* 16, 510 (1996), 140–152.

Vladimir Kolmogorov. 2006. Convergent tree-reweighted message passing for energy minimization. *IEEE Trans. Pattern Anal. Mach. Intell.* 28, 10 (2006), 1568–1583. DOI:http://dx.doi.org/10.1109/TPAMI.2006.200

Yinxia Liu, Weidong Zhou, Qi Yuan, and Shuangshuang Chen. 2012. Automatic seizure detection using wavelet transform and SVM in long-term intracranial EEG. *IEEE Trans. Neural Syst. Rehabil. Eng.* 20, 6 (2012), 749–55. DOI:http://dx.doi.org/10.1109/TNSRE.2012.2206054

David J.C. Mackay. 2003. *Information Theory , Inference , and Learning Algorithms*,

R.J. McEliece, D.J.C. MacKay, and J.F. Cheng. 1998. Turbo Decoding as an Instance of Peal's ``Belief Propagation'' Algorithm. *IEEE J. Sel. Areas Commun.* 16, 2 (1998), 140–152. DOI:http://dx.doi.org/10.1109/49.661103

Talya Meltzer, Amir Globerson, and Yair Weiss. 2009. Convergent message passing algorithms - a unifying view. *Engineering* (2009), 393–401.

KP Murphy, Y. Weiss, and MI Jordan. 1999. Loopy belief propagation for approximate inference: An empirical study. *Proc. Fifteenth …* (1999).

Thiago M. Nunes, André L. V Coelho, Clodoaldo A.M. Lima, João P. Papa, and Victor Hugo C. De Albuquerque. 2014. EEG signal classification for epilepsy diagnosis via optimum path forest - A systematic assessment. *Neurocomputing* 136 (2014), 103–123. DOI:http://dx.doi.org/10.1016/j.neucom.2014.01.020

R. Panda et al. 2010. Classification of EEG signal using wavelet transform and support vector machine for epileptic seizure diction. *2010 Int. Conf. Syst. Med. Biol.* , December (2010), 405–408. DOI:http://dx.doi.org/10.1109/ICSMB.2010.5735413

Judea Pearl. 1988. Probabilistic Reasoning in Intelligent Systems. *Morgan Kauffmann San Mateo* 88 (1988), 552. DOI:http://dx.doi.org/10.2307/2026705

L.R. Rabiner. 1989. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proc. IEEE* 77, 2 (1989), 257–286. DOI:http://dx.doi.org/10.1109/5.18626

Sriram Ramgopal et al. 2014. Seizure detection, seizure prediction, and closed-loop warning systems in epilepsy. *Epilepsy Behav.* 37 (2014), 291–307. DOI:http://dx.doi.org/10.1016/j.yebeh.2014.06.023

Alexis Roche. 2011. EM algorithm and variants: an informal tutorial. *Spring* (2011), 1–17.

Saeid Sanei and Jonathon. a. Chambers. 2007. *EEG Signal Processing*,

Solomon Eyal Shimony. 1994. Finding MAPs for belief networks is NP-hard. *Artif. Intell.* 68, 2 (1994), 399–410. DOI:http://dx.doi.org/10.1016/0004-3702(94)90072-8

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Matteo Causo, Lille 1, 2017

Ali Shoeb and John Guttag. 2010. Application of Machine Learning To Epileptic Seizure Detection. *Proc. 27th Int. Conf. Mach. Learn.* (2010), 975–982. DOI:http://dx.doi.org/10.1016/j.jneumeth.2010.05.020

Alexandros T. Tzallas, Markos G. Tsipouras, and Dimitrios I. Fotiadis. 2009. Epileptic seizure detection in EEGs using time-frequency analysis. *IEEE Trans. Inf. Technol. Biomed.* 13, 5 (2009), 703–710. DOI:http://dx.doi.org/10.1109/TITB.2009.2017939

Martin J. Wainwright, Tommi S. Jaakkola, and Alan S. Willsky. 2003. Tree-based reparameterization framework for analysis of sum-product and related algorithms. *IEEE Trans. Inf. Theory* 49, 5 (2003), 1120–1146. DOI:http://dx.doi.org/10.1109/TIT.2003.810642

Y. Weiss and W.T. Freeman. 2001. On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs. *IEEE Trans. Inf. Theory* 47, 2 (2001), 736–744. DOI:http://dx.doi.org/10.1109/18.910585

Yair Weiss. 1997. Belief propagation and revision in networks with loops. *A.I. Memo* , 1616 (1997).

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Chapter 6 Conclusions and Future Works

In the present PhD thesis work we target two ambitious objectives: (i) approaching Neuro-Inspired computing with an Ultra-Low-Power (ULP) prospective and (ii) simplifying and enhancing Machine Learning algorithms, so that to bring intelligent processing as close as possible to data sources and to popularize Machine Learning over strictly constrained electronics such as portable, wearable and implantable devices.

In other words, we aim at building a Neuro-Inspired ULP HW that closely meets the Hierarchical Temporal Memory (HTM) principles and theory introduced in Section 1.3.4. It has to be general purpose, capable of on-line learning, continuous inferring and predicting. It has to start without any knowledge on its inputs and the elaboration it has to perform, rather it learns everything from data and continuously adapts its internal representation of the external non-stationary world. Complexity and hierarchy have to be automatically and naturally incorporated by a top-down application-driven *conscious* and *contentious* design.

To attain these ambitious objectives, we formulate and develop the basic idea of exploiting the exceptional abilities of Bayesian Machine Learning (BML). In fact, the BML covers and manipulates inference, prediction, on-line learning and includes tools for sequential data analysis as required in the HTM theory. Yet the intimate working principles of the human brain are not completely known, but advanced BML algorithms like the Variational Bayesian Expectation-Maximization (VBEM) are able to learn the model structure on-the-fly, resembling the brain's ability to plastically shaping its structure. Figure 6:1 follows from Figure 1:5 and illustrates this concept.

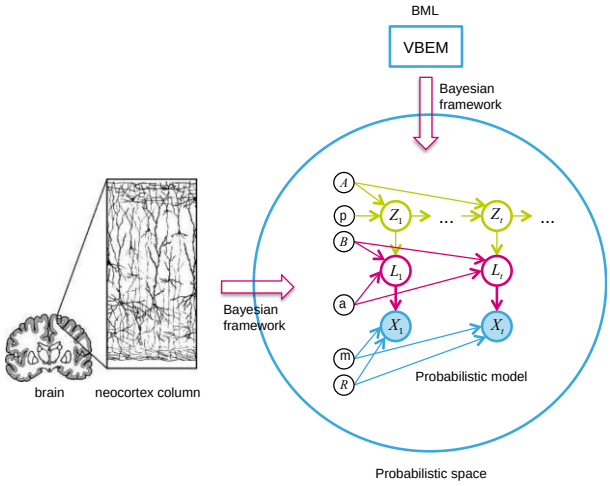


Figure 6:1 Main idea behind this PhD thesis revised.

The principal takeaway of Figure 6:1, is that the Bayesian framework acts as an interface that projects the human brain working mechanisms into the probabilistic space and represents them as a continuously-evolving plastic probabilistic model of the external world. Through BML, we can access to this same probabilistic space and learn and manipulate a similar model to that the human brain employs. Therefore, the present work focuses on the arrow connecting the BML to the Hidden Markov Model Gaussian Mixture Model (HMM GMM) implicit in the probabilistic space. We have chosen a HMM GMM as generic structure, because we mainly target IoT and medical applications generally related to stream analytics.

BML algorithms and their execution demand for optimizations in order to tackle Big Data challenges. In fact, the state-of-the-art VBEM algorithm in [Ji et al. 2006] has been introduced as an interesting learning solution for HMM GMM parameters and structure. However, its naïve HW implementation results neither effective nor feasible because of the required memory, computing power and overall complexity. In particular, data structures encode redundant information and some parameters depend linearly on the number of input data H, which goes to infinity in streaming applications.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Matteo Causo, Lille 1, 2017

In this work, we developed a less complex on-line, distributed and nonparametric version of the state-of-the-art VBEM HMM GMM. We eliminated the dependence on H by introducing the Greedy Forward Pass, and gain two orders of magnitude in complexity reduction through only algorithm level considerations and optimizations. As a result, a VBEM HMM GMM implementation in hardware becomes both feasible and attractive for wearable, portable and implantable electronics.

Moreover, we studied the spatial distribution for the proposed Algorithm 2 and drew important considerations regarding its noise reduction properties. We derive the paramount relation in Equation 3:11 to trade-off distribution and complexity by achieving a valid final classification accuracy. We then tested our approach on both synthetic and real-world applications, namely the MNIST handwritten digits recognition and EMG-based gesture recognition. We generally show better results with respect to the state-of-the-art solutions.

Target applications, however, result independent on their actual implementation because the probabilistic model, BML creates, is implicit and populates the probabilistic space. In fact, the proposed algorithm can be mapped on conventional CMOS HW, where further optimization techniques can be applied. We demonstrated this point by designing a proof-of-concept HW implementation on FPGA platform for real-time streams analytics. In particular, this led to the following outcomes:

- We validate our choice for the BML by demonstrating it leads to convenient and attractive optimizations even without the need to migrate towards a different technology than the CMOS. In fact, it results more suitable than other Neuro-Inspired approaches to Learning-On-a-Chip solutions such as Forward Neural Networks and kernel methods. Furthermore, BML is a better choice for IoT and medical applications, because it performs generally better than dedicated accelerators for Deep Learning and supervised Supporting Vector Machines.

- We define a dedicated solution for the digamma function $\psi(\cdot)$ that is hard to compute analytically and it is generally approximated even in software-based implementations. We conveniently exploit the on-line elaboration to obtain an incremental/decremental digamma calculator. With this same principle, we build on an incremental/decremental reciprocal calculator. Both contributions represent an innovation with respect to the state-of-the-art.

- We further optimize the Greedy Forward Pass by operating in Log-domain. This results in an optimized matrix multiplication technique.

- We finally build on an advanced setup to flexibly test our HW implementation on a FPGA platform and produce three real-time demos on the same applications of interest as in Section 3.7.

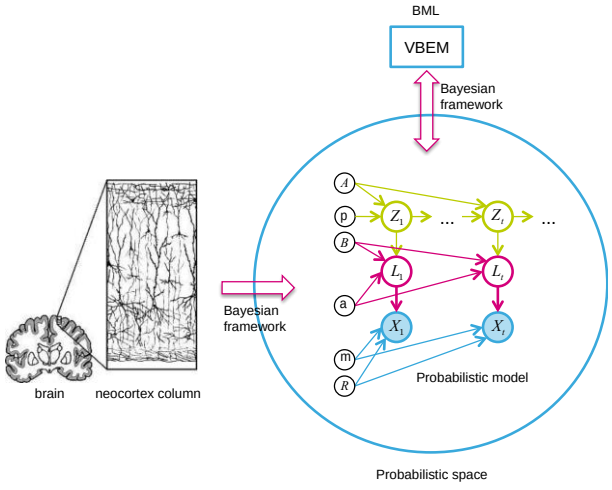


Figure 6:2 Main idea behind the present PhD thesis unabridged.

These are important results in two ways. Firstly, we build on a proof-of-concept that validates our thesis and meets its main objectives. Secondly, we demonstrated that we are able to realize in low-power hardware an advanced learning algorithm, having the ability to learn and build any model of the reality in a plastic way, similarly to as the human brain. Certainly, we do not employ the sophisticated techniques of the brain to shape its structure, because we do not know them yet. In fact, we exploit the Bayesian model selection among infinite significant models rather than constructively building the true one from data. Nevertheless, we demonstrate no matter the underlined model of the reality, we are able to summarize the ultimate findings in Machine Learning into

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Matteo Causo, Lille 1, 2017

a generally valid algorithm that can be implemented in hardware and optimized for strictly constrained applications. In other words, a further major contribution of the present PhD thesis work consists in making bidirectional the arrow that connects the BML to the probabilistic model into the probabilistic space, as shown in Figure 6:2.

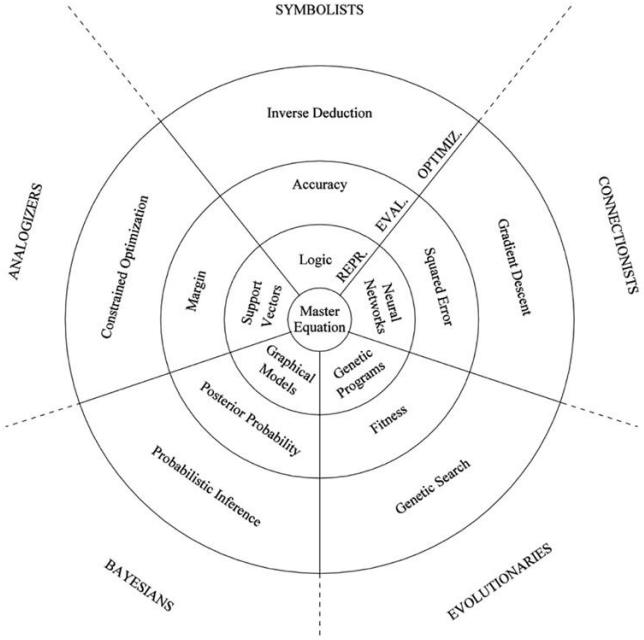


Figure 6:3 Schematic diagram for each of the tribes.

(The Master Algorithm, Pedro Domingos, got from Book Summary)

The arguments made here resonate very similar to the central hypothesis expressed by Pedro Domingos in his recent book *The Master Algorithm: How the Quest for the Ultimate Learning Machine Will Remake Our World*. He states "All knowledge – past, present, and future – can be derived from data by a single, universal learning algorithm". Indeed, he speculates the presence of a master algorithm and says that there are arguments from many fields, namely Neuroscience, Evolution, Physics, Statistics and Computer Science that speculate about it. This idea is nothing new. In fact, the HTM theory, which we refer to, assumes that homogeneity in the neo-cortex structure suggests the human brain performs the same algorithm everywhere, so that the model it employs may be generic to cope with a number of different applications.

Tribe	Origins	Master Algorithm
Symbolists	Logic, philosophy	Inverse deduction
Connectionists	Neuroscience	Backpropagation
Evolutionaries	Evolutionary biology	Genetic programming
Bayesians	Statistics	Probabilistic inference
Analogizers	Psychology	Kernel machines

Figure 6:4 Tribes, origins and relative master algorithm.

(The Master Algorithm, Pedro Domingos, got from Book Summary)

Nevertheless in his book, Domingos gives a complete and comprehensive picture of the ML field illustrated in Figure 6:3. He divides ML algorithms into five categories he refers to as tribes. Then, he explains the rationale of a possible Master Algorithm by taking the reader through a tour of each of the tribes' philosophy and their own master algorithms and comes up a unifier, called Alchemy, which he refers to as the Master Algorithm. In fact, each tribe has its own master algorithm for prediction that it is not generic, but specific to some type of problems, as shown in Figure 6:4.

The symbolists' tribe believes all intelligence can be reduced to manipulating symbols and learning to the *inverse of deduction* according to predefined rules. Evolutionaries focus on *learning the structure*. In contrast, connectionists assume a simple, hand-coded structure with lots of connections and let back-propagation *learning the weights*. They reverse engineer the brain and are inspired

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

by neuroscience and physics. For Bayesians, knowledge goes in the prior distribution over the structure and parameters of the model. They believe that learning is a form of *probabilistic inference* and have their roots in statistics. Bayesians and symbolists agree that prior assumptions are inevitable, but they differ in the kinds of prior knowledge they allow. Finally, the Analogizers tribe learns by extrapolating from *similarity* judgments and is influenced by psychology and mathematical optimization. This tribe uses similarities among various data points to categorize them in to distinct classes.

The Master Algorithm is a single algorithm combining the key features of all of the tribes' master algorithms and identifies the intelligent learning core. In fact, Domingos criticizes the representatives of each tribe because they firmly believe that theirs is the only way to model and predict. Unfortunately, this thinking hinders their ability to model a broad set of problems[16].

Similarly, the present PhD thesis work is cross-sectorial in ML. Indeed, the proposed on-line VBEM HMM GMM is capable to combine most of the key features of all tribes. Nevertheless, the ultimate aim of this thesis is not to present the VBEM as the Master Algorithm. We are convinced the Bayesian approach gathers in their entirety the brain's mechanisms described in the HTM theory, but it does not necessary encode the real behavior of the brain. Therefore, the BML can be employed to study and better approximate the Master Algorithm as future work.

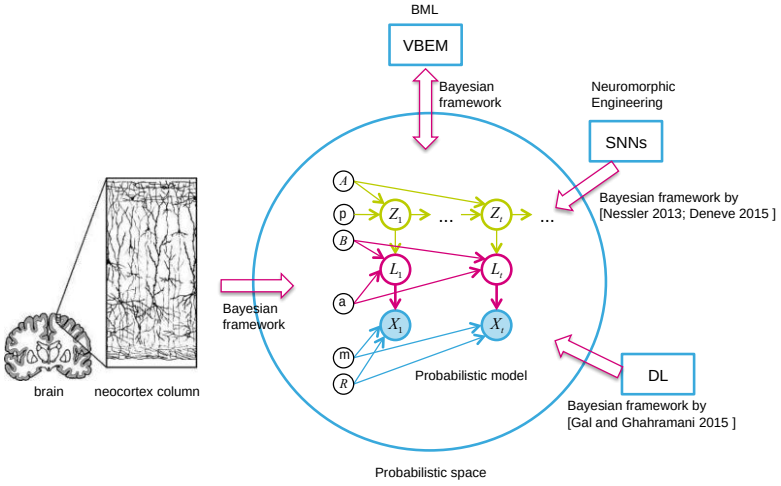


Figure 6:5 Main idea behind this PhD thesis extended to future developments.

Anyhow, very recent and encouraging developments consistently link Deep Learning (DL) [Gal and Ghahramani 2015b; Gal and Ghahramani 2015a; Ghahramani 2015] and learning and inferring mechanisms in Spiking Neural Networks (SNNs) [Nessler et al. 2013; Deneve 2015; Pecevski and Maass 2016] to BML. As depicted in Figure 6:5, Neuromorphic Engineering and in particular SNNs can be explained through the Bayesian framework, as projecting the observed world into a probabilistic model that evolves during the continuous learning. So is for the DL. Therefore, for both DL and Neuromorphic Engineering the arrow toward the probabilistic model has been recently established. However, as explained in Section 1.3.1, DL and Neuromorphic Engineering exhibit the same basic problem: profound principles of brain functioning are not yet known, therefore *conscious* or *conscientious* network design is not yet possible. In particular, the DL design follows the "*bigger is better*" rule of thumb, with no room for optimizations with respect to the low-power objective. Conversely, the Neuromorphic Engineering approach intrinsically proceeds bottom-up, without an overall application-driven view.

Our idea for future developments is to reproduce the work done in this PhD thesis, by developing bidirectional arrows connecting the Neuromorphic Engineering and DL to the probabilistic model. In this way, we would be able to derive the design rules for a *conscious* or *conscientious* design by starting from a given probabilistic model suitable for the most various applications. In particular, we can employ BML to accurately define the resources and the system to build. In this way, we will have a punctual control of the neuro-inspired HW and a-priori understanding of each element function. Indeed, our approach to Neuro-Inspired will be *top-down* and will make any kind of *conscious* and *conscientious* optimizations feasible.

These ultimate goals represent disruptive advances in the fundamental AI and ML research with substantial repercussions in the technological progress. We combined a visionary research with the real market needs of a product-oriented company. Following such

[16] The Master Algorithm: Book Summary.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Matteo Causo, Lille 1, 2017

route, short-term objectives may be to robustly validate our findings and achievements through a number of real-world practical applications, such as real-time health monitoring for the Human Intranet [Rabaey 2015].

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Thèse de Matteo Causo, Lille 1, 2017

References

Sophie Deneve. 2015. Bayesian Inference with Spiking Neurons. In *Encyclopedia of Computational Neuroscience*. 361–364. DOI:http://dx.doi.org/10.1007/978-1-4614-6675-8_568

Yarin Gal and Zoubin Ghahramani. 2015a. Bayesian Convolutional Neural Networks with Bernoulli Approximate Variational Inference. (2015), 12. DOI:http://dx.doi.org/10.1029/2003JD002581

Yarin Gal and Zoubin Ghahramani. 2015b. On Modern Deep Learning and Variational Inference. *Adv. Approx. Bayesian Inference Work. NIPS* (2015), 1–9.

Zoubin Ghahramani. 2015. Probabilistic machine learning and artificial intelligence. *Nature* 521, 7553 (2015), 452–459. DOI:http://dx.doi.org/10.1038/nature14541

Shihao Ji, Balaji Krishnapuram, and Lawrence Carin. 2006. Variational Bayes for continuous hidden Markov models and its application to active learning. *IEEE Trans. Pattern Anal. Mach. Intell.* 28, 4 (2006), 522–532. DOI:http://dx.doi.org/10.1109/TPAMI.2006.85

Bernhard Nessler, Michael Pfeiffer, Lars Buesing, and Wolfgang Maass. 2013. Bayesian Computation Emerges in Generic Cortical Microcircuits through Spike-Timing-Dependent Plasticity. *PLoS Comput. Biol.* 9, 4 (April 2013), e1003037. DOI:http://dx.doi.org/10.1371/journal.pcbi.1003037

Dejan Pecevski and Wolfgang Maass. 2016. Learning probabilistic inference through STDP. *eneuro* (2016), ENEURO--0048.

Jan M. Rabaey. 2015. The human intranet - Where swarms and humans meet. *IEEE Pervasive Comput.* 14, 1 (2015), 78–83. DOI:http://dx.doi.org/10.1109/MPRV.2015.20

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Chapter 7 Appendix

7.1 Extracting invariant features using auto-encoder by Etienne MARECAL

In the industrial problematics as much as in a research context, **Deep Learning (DL)** has allowed **Artificial Neural Network**s **(ANN)** to succeed in many image recognition and classification problems where predictions should be unchanged, or invariant, under one or more transformations of the input variables. For instance, a dangerous pedestrian should be detected as such by the on-board computer of an automotive car whether he is tall or small, dressed in red or blue, in front, side or back view, etc. All these example are transformations of the input data that should leave the conclusion unchanged. DL algorithms has proven that they can learn and generalize this invariance property and have allowed ANNs to reach high classification performances. The problem is that DL requires huge computational resources; a problematic that follows is thus to think of alternatives, as efficient but less resources-consuming.

Invariance can indeed also be built into a separate preprocessing engine by extracting features that are invariant under the required transformations, so that DL could be replaced by a lighter and simpler **Machine Learning (ML)** algorithm in order to perform the classification. Therefore, invariance principles inspired by DL could lead an efficient external **Feature Extraction Engine**s **(FEE)** to fit into portable and wearable electronics with limited resources constraints.

7.1.1 Research work on feature extraction engines

The work presented in the following is the result of the research on FEE able to extract invariant features. It should be read as an exploration of diverse assumptions in order to understand the behavior of certain types of machine learning algorithms used to proceed feature extraction. With the restricted amount of time dedicated to the project, the goal was not to prove mathematically these hypotheses but rather to get familiar with the capabilities of an ANN, its limits and its particularities, in order to be able to apply this understanding to the design of a new algorithm, that could outperform basic algorithms on which it is based.

In order to determine the performances of the FEEs designed, we selected a 'toy-example'. The algorithms were tested on the classification task of the handwritten digits of the MNIST dataset (classical dataset to evaluate ML algorithms efficiency). The data were pre-processed by our algorithms to extract features from them. The features, stacked in what we name a **feature vector (FV)**, were then used to proceed to classification thanks to standard classification algorithms. Our main purpose by designing the FEE was then to maximize the accuracy rate of the classification performed using the features.

7.1.2 Why extract invariant features?

An assumption that initiated this work on feature extraction can be stated as "*good feature extraction is obtained when highly invariant features are extracted from the data*". Let's defined more precisely concepts such as "good feature extraction" and "highly invariant features".

"Good feature extraction" means in this work that the data processed will be relevant to process the classification task. However, the accuracy rate depends on the classifier used. There exist different types of classifier: roughly, linear ones and more complex ones. The second type leads in most cases to better accuracy but is more resource-consuming than the first. Keeping our goal in mind, expressed in the introduction, the point of this work is precisely to use the first type of classifier, and using the pre-processing of data, to reach the performances of the second type. That is why classifier of both types will be used and compared in the following.

Different types of feature extraction could be defined. Some are known from the literature, others were experimental. We used in this work:

- Unsupervised algorithms: auto-encoders (AE) and stacked auto-encoders (up to 2 levels). The feature vector in that case is the central layer of the (last) auto-encoder. We will use respectively the names 'AEnb' and 'stacked AEnb1-nb2' when presenting our results. nb, nb1 and nb2 represent the number of neurons in the central layer.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

- Supervised algorithms: we used a Convolutional Neural Network (CNN) (known for its high performances in the case of image classification). In this case we chose to use the last layer before the output layer as a feature vector: its values are the result of the different operations of convolution and pooling on the image. This use of the CNN was one of the experimental design of a feature extraction engine.
- We also used an algorithm that derivate from an auto-encoder, that we name a 'representative auto-encoder'. It has the same structure of an AE, but instead of making the algorithm learn to reconstruct the input data in the output layer as it is traditionally done, we tried in this design to reconstruct what we name a 'representative element' of the class to which the current input data belongs to (we have elected such an element for each 10 classes of data of the MNIST dataset).

Using the FEE to process data, we get a new dataset composed of all the feature vectors extracted. Let's focus on one of the component of the FVs. If, for instance, it as a value very similar for all the handwritten '2' that present a certain characteristic, we could consider that this component represents this particular characteristic, that we name a feature. Then, independently of how big the '2' is drawn, how rotated it is, how thick the line is, the component should have an approximately equal value for all the '2' that present this particular feature. If this value is equal for most '2', the feature would be considered 'invariant', otherwise it would be a 'variant' component.

To be clearer on what we consider as 'approximatively equal': we have chosen that if for more than half of the data from one class, the feature presents a value that is included in a 0.05 interval centered on the average value (0.05 represent 5% of the range of value that the feature can take: they are ranging between 0 and 1), then this feature is considered as invariant. Then, as this definition of invariance enables us to decide whether or not a feature is invariant for a particular class, we can count for this class the number of features that are invariant. We divide this result by the total number of features to get what we name the 'invariance rate' for the class we are considering.

Let's now average the invariance rate over all the class of the dataset: we get a global average invariance rate that reflects the invariance the FEE have extracted from the data. We name this indicator the '**invariance rate of the FEE**'. We could discuss these choices: why half of the values, why 5%, why simple arithmetic mean, but this is not the point: we assume that such a definition is sufficient to observe, even roughly, the quantity of invariance we can extract from the data, and to use this information to highlight the correlation expected between invariance and classification accuracy.

The first step of this research work was to verify to which extent our assumption that invariant features lead to efficient classification is true. Thus, we designed and used different FEEs that can extract different rate of invariance from the raw data. Once we have these FEEs, we get transformed datasets with which we can proceed classification. The results we got are presented in the following table. Confirming our assumption would then resume in showing that the accuracy of classification is a growing function of the invariance rate.

Classification rate (%) **obtained using matlab's classification library tools**:					Invariance rate (%) obtained with each FFE
Input	**Linear SVM**	**Quadratic SVM**	**Cubic SVM**	**Medium Gaussian SVM**	**Invariance**
CNN	85.8	86.5	86.5	86.2	**4.5**
Stacked AE400-100	**95.1**	**97.5**	97.7	97.3	**9.9**
AE100	93.9	**97.5**	**97.8**	**97.4**	**11.8**

Figure A:1 - Correlation between invariance rate and classification accuracy obtained for different FEEs

We can observe that indeed classification accuracy seems roughly to grow with invariance rate and that it depends heavily on the classifier used.

Our first assumption seems to be confirmed, yet we designed the following experimentation in order to confirm this trend while focusing on a particular FEE.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

In this experiment we used a 3-layers auto-encoder: input and output layer have the same number on neurons which depends on the data; the middle layer is the feature vector: we make its size vary which makes the invariance rate vary. Figure A:2 presents in the middle a graph showing the evolution of the invariance rate when the size of the FV varies; on the right it shows the accuracy of classification obtained using the previews feature vectors (using a Linear SVM classifier whose parameters are set to default values by MatLab toolbox). The shapes of each of these curves could be commented but the point here is more to compare these curves together.

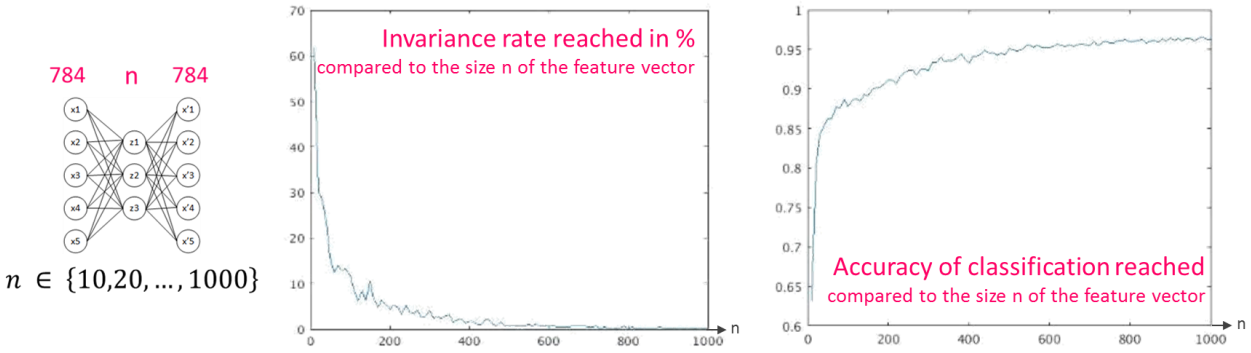


Figure A:2 - (left) Structure of AE used, (middle) invariance rate extracted from the data using the AE, (right) accuracy of classification reached using the features previously extracted

What appears first is that the trend between the invariance rate extracted and the classification accuracy is contrary to what we expected. We interpret this as follows: the influence of the FV's size on the classification accuracy is greater than the influence of the invariance rate. And indeed, here we could not isolate the influence of the invariance rate on the classification accuracy, we are rather showing the influence of the size of the feature vector which varies from 10 to 1000. This experiment helps understanding FEEs and is worth mentioning, but the following is more interesting regarding our problematic.

In this experiment we designed a FEE that we could tune in order to make the invariance rate vary, without making the size of the feature vector vary. We use a stacked auto-encoder composed of 2 auto-encoder: the middle layer of the first one is considered as input of the second auto-encoder. We fixed the size of the middle layer of the second AE, which is the feature vector, while we made the size of the middle layer of the first AE vary. Using this design, we simulated diverse FEEs with different invariance rate, without making the size of the feature vector vary. We then used the transformed data we obtained to proceed to classification. Figure A:3 shows the evolution of classification rate when invariance rate varies (using a Gaussian Naïve Bayesian classifier).

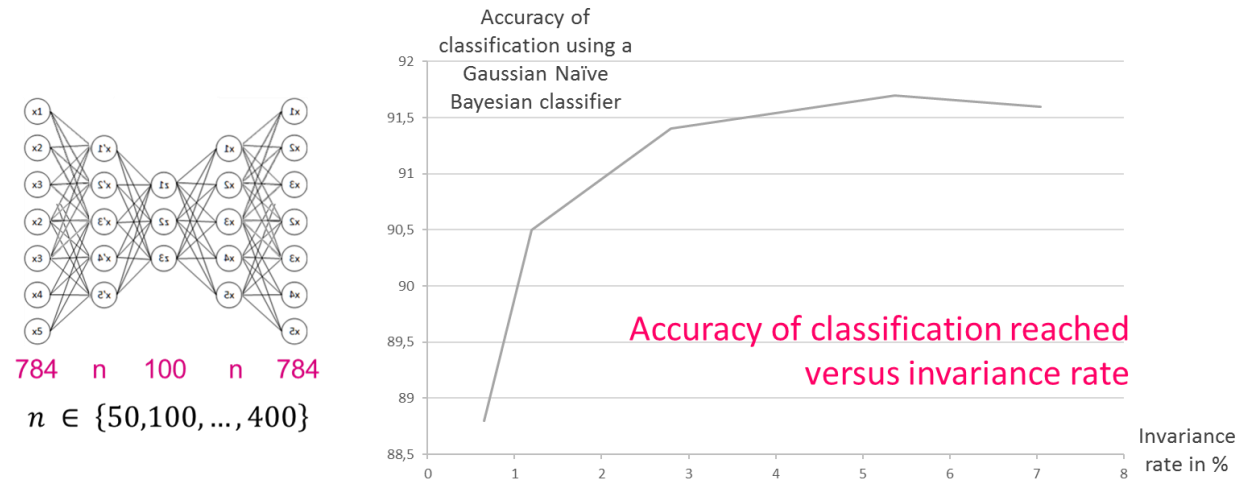


Figure A:3 - (left) Structure of stacked AE used, (right) accuracy of classification reached versus invariance rate of the features

We could reproduce this growing curve with different size of the FV. Thus, we can conclude that, once we are able to isolate the effect of the invariance rate, the get the expected result: classification accuracy is a growing function of the invariance rate.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

The final point of this works is to use a simple classifier (we chose a naïve Gaussian Bayesian classifier) paired to our FEE in order to reach classification accuracy at least as good as those reached with complex classifier (such as SVM for instance). We should then design and tune the FEE to maximize the invariance rate so that the classification accuracy is maximized.

We designed the 'representative auto-encoder' described below. This particular algorithm turned out to be the algorithm with which we could extract the higher rate of invariance (almost up to 50% of invariance according to our criteria). Figure A:4 shows a comparison of different algorithms and present the invariance rate that could be extracted from the raw data and the classification accuracy obtained in each case.

	Invariance rate (%) obtained with each FFE:	Classification rate (%) obtained using matlab's classification library tools and computation time associated:		
Input	**Invariance rate**	**Gaussian SVM**	**Linear SVM**	**Naïve Gaussian Bayesian**
Raw input	2.5	11.1	93.8	75.4
Stacked AE400-100	9.6	16.9	95.7	84.3
AE100	11.8	15.9	94.2	84.2
Representative AE100	28.7	56.9	97.4	94.5
Representative AE400-100	48	97.6	97.7	97.6

Figure A:4 - invariance rate reached with diverse FEEs and classification accuracy obtained with these features

The greater result on which we want to focus is the fact that with this process, using this simple classifier we could progress from a quite low classification accuracy (75.4% using raw data) to a classification accuracy significantly better and at the level of what could be done with a more complex classifier as a Linear SVM (97.6% using the 'representative auto-encoder', compared to 97.6% for the linear SVM).

7.1.3 Future works

Few issues are worth being considered, yet. First, based on our current algorithms, we could design a new one that would maximize invariance of the FV during the training phase (for instance, by introducing a term in the cost function of the auto-encoder that penalizes small invariance). It would also be worth precising the definition of invariance while doing this optimization.

Then, a natural step that follows this work is to stress the performances of the last algorithm presented (representative auto-encoder associated with naïve Gaussian Bayesian classifier) with other classification tasks and other datasets. We truly believe that it would export satisfyingly too many tasks.

Finally, to step further in the direction of low consummation it would be interesting to continue this work in a direction that would not use learning, which is resources consuming, in the process of feature extraction.

7.1.4 References

- Yann LeCun, Yoshua Bengio & Geoffrey Hinton, Deep learning
- Honglak, Grosse, Rajesh, Ng, Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations.
- S. Mallat 2016, Understanding Deep Convolutional Networks, Comm. in Philosophical Transactions A.
- Radford, Metz, Chintala, Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks
- Bengio, Learning Deep Architectures for AI
- Le, Zou, Yeung, Ng, Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis
- Lee, Grosse, Ranganath, Ng, Convolutional Deep Belief Networks for Scalable
- Goroshin, Bruna, Tompson, Szlam, Eigen, LeCun, Unsupervized Feature Learning from Temporal Data
- Socher, Huval, Bhat, Manning, Ng, Convolutional-Recusive Deep Learning for 3D Objects Classification
- Zou, Zhu, Ng, Yu, Deep Learning of Invariant Features via Simulated Fixations in Video

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

7.2 Wavelet Scattering applied to Speaker Diarization and EEG Analysis by Artiom TCHOUPRINA

The initial task of the internship was to address the Speaker Diarization problem: segmenting an audio recording of a meeting into different segments corresponding to different speakers. In a nutshell, answering the question: "Who spoke when?" State of the art provides many ways of solving this problem, Gaussian Mixture Models (GMM) can be used to model each speaker, powerful statistical tools such as Hierarchical Dirichlet Process (HDP) and Hidden Markov Models (HMM) are also used. Artificial Neural Networks (ANN) can also be trained to realize clusters corresponding to different speakers.

In all these methods, one important point is the fact that the data to be analyzed is time series, which means that a window is applied to the signal and successive data segments are treated sequentially, with a possible overlapping of a segment with the next one. Since all the previous mathematical tools can be studied very deeply, we chose to focus on this window instead. The goal of the internship was to study the influence of the size of the window and try to find objective criteria to help choosing the size that maximizes classification accuracy.

Once the windowing problem addressed and solved, the goal was to use the same methods and to apply them to another problem: epileptic seizures detection with Electroencephalogram (EEG) analysis. Indeed, a first assumption was that both speech and EEG signals were alike and that detecting a speaker change was similar to detecting a seizure.

We then started to try to adapt part of Stephane Mallat's team work to the project. He has widely contributed to the wavelet theory development. This mathematical framework is very well suited for the project. Indeed, the signals analyzed are non-stationary, which requires a time-frequency analysis. Furthermore, we do not have prior information on the size of the patterns we try to identify in the signals, thus a multi-scale analysis is mandatory, to manage to extract invariant features from the data. Finally, we demand that the operation remains stable to certain time related transforms, such as shifting and warping, this condition seems obvious, the instant when a speaker speaks must not interfere with his identification.

The process used to extract features from the data while satisfying these conditions is called Wavelet Scattering. It consists in Cascading "wavelet modulus" operators and averaging. At each stage, the signal is convoluted against a bank of wavelets shifted and warped in frequency. For mathematical reasons, working with wavelet coefficients requires introducing a nonlinear operator in order to extract invariance, thus a modulus is applied at each stage.

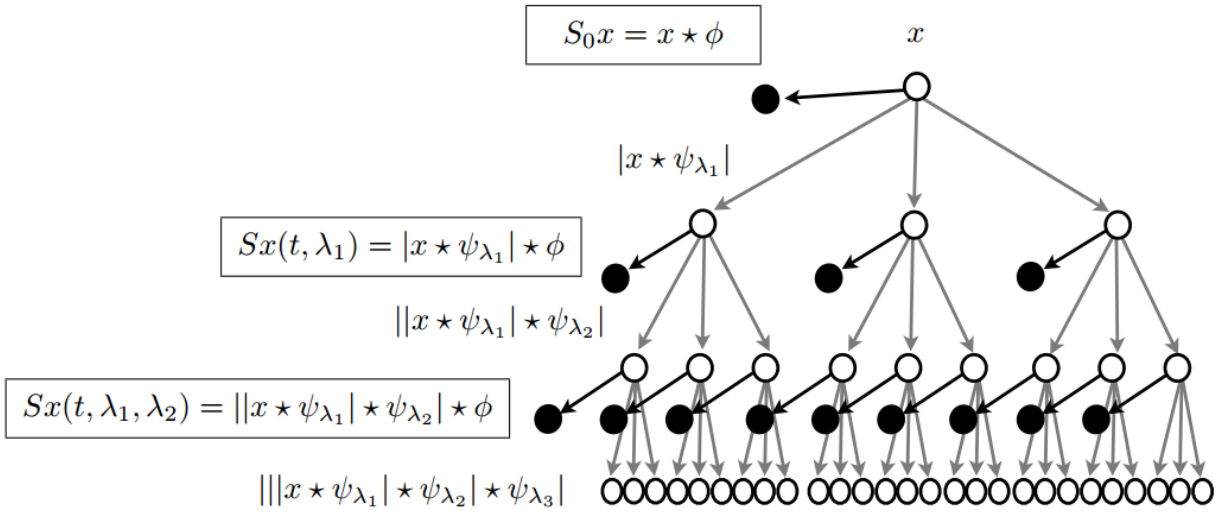


Figure A:5. Wavelet Scattering.

Wavelet Scattering is a rather complex process compared to other feature extraction engines. In its construction, cascading averaging, nonlinear operators and convolutions, it features the same transformations that are included in a Convolutional Neural Network (CNN). The difference with CNNs is that there is no learning here, the filter banks are fixed from an early stage, whereas a neural network learns and adapts its coefficients during the learning stage.

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

Mallat's team has developed a MATLAB Toolbox that features all the tools needed for wavelet transforms. This toolbox, ScatNet, allows managing databases, creating filter banks, doing and visualizing transforms. It realizes all the steps of the scattering and allows a good representation of the results.

Once the coefficients are extracted, the classification can be done with various methods From K Nearest Neighbors (K-NN), to Support Vector Machines (SVM), the only constraints is how much resources is available for the calculation. Yet, K-NN is excluded since it requires knowing the number of clusters required.

The size of the time averaging window can have an influence on the information extracted. Other transforms cannot handle too large windows, for example a Sliding Discrete Fourier Transform (SDFT) does not allow a window larger than 20ms for sounds, and thus larger patterns are not detected. The advantage of Wavelet Scattering is that these large patterns are extracted; in addition, transients, attacks, and other shorter information are detected as well. With a larger window the energy is located deeper in the layers of scattering. One wavelet scattering is equivalent to several other transformations with different time windows, the level of details to which we have access corresponds to the layer which is observed. This is illustrated in the Table A:1 and Table A:2, on two different datasets, speech and music, depending on the type of data, the energy is not located evenly in the layers, a different choice of windowing size is relevant for these two sets, and we believe that the distribution of energy in the layers gives a criterion for the choice of the window size.

Table A:1: Energy location vs scattering order for different averaging window size for the TIMIT Database (Andén, J (2014) Deep Scattering Spectrum)

Window size / number of scatterings	M=0	M=1	M=2	M=3
T=23ms	0,0%	94,5%	4,8%	0,2%
T=93ms	0,0%	68,0%	29,0%	1,9%
T=370ms	0,0%	34,9%	53,3%	11,6%
T=1,5s	0,0%	27,7%	56,1%	24,7%

Table A:2: Energy location vs scattering order for different averaging window size for GTZAN Database

Window size / scattering order	M=0	M=1	M=2	M=3
T=23ms	1.32%	90.12%	4.70%	0.16%
T=93ms	0.01%	87.47%	10.83%	0.68%
T=370ms	0.00%	69.82%	25.24%	2.99%
T=1,5s	0.00%	54.68%	34.58%	7.74%

© 2017 Tous droits réservés.

lilliad.univ-lille.fr

The Database that is classically used for the Speaker Diarization task (TIMIT) is not open source, another database was used to test performance of the speaker diarization. The task was to classify each sound into one of the 10 musical genres available. This Database (GTZAN) was classified with an 84.3% accuracy with a Gaussian Support Vector Machine classifier.

Another Database that Wavelet Scattering was applied on is an Electroencephalogram (EEG) database. The task is to detect moments when the patient has an epileptic seizure. This database contains hours of monitoring of 23 patients, some of whom have around forty sessions of several hours each time. Processing all of this data and apply wavelet scattering represents a lot of computation, make the dataset adaptable to the toolbox is among future goals.

As future work, can be mentioned proving that the energy repartition gives the right criterion for the choice of the averaging window size. Making the ScatNet wavelet scattering toolbox work along with the EEG Database and managing to make the entire process work on-line.

7.2.1 References

- Mallat, S. (2012). Group invariant scattering. Communications on Pure and Applied Mathematics, 65(10), 1331-1398.
- Andén, J., & Mallat, S. (2014). Deep scattering spectrum. IEEE Transactions on Signal Processing, 62(16), 4114-4128.
- Sifre, L., & Mallat, S. (2013). Rotation, scaling and deformation invariant scattering for texture discrimination. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 1233-1240).
- Andén, J., & Mallat, S. (2011, October). Multiscale Scattering for Audio Classification. In ISMIR (pp. 657-662).
- Andén, J., Lostanlen, V., & Mallat, S. (2015, September). Joint time-frequency scattering for audio classification. In 2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP) (pp. 1-6). IEEE.
- Mallat, S. (2016). Understanding deep convolutional networks. Phil. Trans. R. Soc. A, 374(2065), 2015020

© 2017 Tous droits réservés.

lilliad.univ-lille.fr