Elderly monitoring using decision trees under domain shifts and computational resource constraints

Mounir Atiq

université
PARIS-SACLAY

Elderly monitoring using decision trees under domain shifts and computational resource constraints

Monitoring de personnes âgées basé sur des modèles d'arbres de décision en conditions de variation de domaines et de ressources computationnelles limitées

Thèse de doctorat de l'Université Paris-Saclay

École doctorale n° 574, École doctorale de mathématiques Hadamard (EDMH)
Spécialité de doctorat: Mathématiques appliquées
Graduate School: Mathématiques, Référent : ENS Paris-Saclay

Thèse préparée dans l'unité de recherche Centre Borelli (Université Paris-Saclay, CNRS, ENS Paris-Saclay), sous la direction de Nicolas VAYATIS, Professeur à l'ENS Paris-Saclay, et la co-direction de Mathilde MOUGEOT, Professeure à l'ENS Paris-Saclay.

Thèse soutenue à Paris-Saclay, le 3 Juin 2022, par

Mounir ATIQ

Composition du jury

Jérôme BOUDY Professeur, Institut Mines-Télécom	Président
Éric CAMPO Professeur, Université de Toulouse - Jean Jaurès	Rapporteur & Examinateur
Fabrice ROSSI Professeur, Université Paris-Dauphine	Rapporteur & Examinateur
Argyris KALOGERATOS Chargé de recherche, ENS Paris-Saclay	Examinateur
Sergio PEIGNIER Professeur, INSA Lyon	Invité
Nicolas VAYATIS Professeur, ENS Paris-Saclay	Directeur de thèse

THESE DE DOCTORAT
NNT : 2022UPASM004

Remerciements

Je remercie très chaleureusement mon directeur de thèse Nicolas Vayatis et ma directrice Mathilde Mougeot pour leur compétence, leur expérience et tous leurs conseils salutaires aux moments où j'en avais besoin. Je veux également les remercier pour leur patience et leur bienveillance qui m'ont permis d'aller jusqu'au bout. Je remercie également tous les membres du jury d'avoir accepté et pris le temps de lire mon manuscrit et d'assister à ma soutenance, en me faisant l'honneur de montrer de l'intérêt pour mon travail.

Merci à tous les collègues et amis du centre R&D Tarkett avec qui j'ai commencé cette thèse en merveilleuse compagnie. En particulier Renan Serra, Suzanne Varet et Richard Peres qui ont fait en sorte que ma thèse se déroule dans les meilleures conditions. Je garderai des souvenirs magnifiques et impérissables de mon passage en Belgique et pour cela je tiens à remercier particulièrement Damien, Sydney, Rémi, Bernardo, Ludo Oud, Lejla, Pauline, Lise, Eleonore, Clémence, Aurélie, Sarah, Margot et Ismaël.

Je veux remercier tous les collègues de thèse que j'ai eu la chance de rencontrer au Centre Borelli qui sont également devenus de bons amis et qui, en plus d'être extrêmement doués, ont un grand coeur et des personnalités adorables et atypiques. Merci donc à Sergio, Charles, Thomas, Juan, Pierre, Batiste, Marie, Mathilde, Alice, Myrto, Guillaume, Théo, Antoine M. et Antoine De M. pour tous ces bons moments passés ensemble au travail ou en dehors et pour toutes ces discussions profondes et fascinantes. Mention spéciale pour mon binôme de thèse Cifre Tarkett: Ludovic Minvielle avec qui nous avons partagé le pire et surtout le meilleur de nos thèses, et qui a su garder son humour, ses taquineries et sa joie de vivre légendaires même dans les moments les plus durs. Merci aussi à Christophe, Agnès, Véronique, Virginie et Alina pour votre aide, votre bonne humeur et votre gentillesse.

Bien sûr un grand merci également à tous mes amis d'école d'ingé qui m'ont toujours encouragé et aidé à tenir le coup, en particulier les frères Tanguy, Camara, et Dave qui répondent toujours présents lorsqu'il s'agit de se réunir depuis l'époque du BDE. Et merci à tous les autres amis de cette belle époque: Mathias, Hammer, Louis, Lorgouillous, Robin, Segovia, Nasra, Marie, Adrien, Lydia, Lauriane, Guigui, Awa, Kenza, Halimi et tant d'autres! Je remercie aussi tous mes amis de jeunesse qui ne m'ont pas oublié malgré la distance et les années qui passent: Antonin, Armaël, Régis, Barbosa, Jérôme, Hugues, Gauthier, Cyril, Tommy, Anthony, Clément, et Manon. J'espère tous vous revoir bientôt.

J'aimerais en outre remercier deux personnes à la sagesse, au courage et au talent exceptionnels: Onora Dafor et Mysa pour avoir contribué sans le savoir à mon apaisement dans les moments les plus difficiles d'une époque tumultueuse.

Enfin, d'une manière plus intime je tiens avant tout à remercier Dieu evidemment ainsi que mes parents qui m'ont permis, par leur intelligence, leur sagesse et leur affection, de ne jamais perdre de vue les repères essentiels dans la vie. Et bien sûr un grand merci à mes soeurs adorées qui ont su me faire voir le meilleur de moi même à travers leurs yeux et grâce à qui j'ai pu développer mon sens des responsabilités. Je remercie aussi toute ma famille au sens large qui m'a toujours choyé et comblé depuis ma naissance, en particulier Abdelhak qui m'a nourri de sa fascination pour les sciences depuis mon enfance.

Contents

Abstract

This thesis was sponsored by Tarkett in the context of CIFRE (Convention Individuelle de Formation par la Recherche) research project on the topic of elderly daily life real-time monitoring and is supported by the Industrial Data Analytics & Machine Learning Chair (IDAML) of ENS Paris-Saclay. Tarkett is a global flooring company that developed a piezo-electric sensor encapsulated in the flooring and an embedded system meant to be equipped in nursing home patient rooms. Their objective through this industrial project is to build reliable machine learning models able to work in real-time in the embedded system, based on piezo-electric signals, to provide useful information for medical staff to monitor their patients health.

We first present Tarkett *Floor in Motion Care* (FIM Care) smartfloor technology and characterize particularities of its piezo-electric signal and how it is processed into time series that are used for A.I. elderly monitoring. Considering different measurement technologies we describe how they affect the original physical signal, as well as different data gathering environments in which several dataset have been recorded. To be able to monitor elderly health state some important recurrent events like walk and some anomalies like falls need to be recognized from floor sensor signals. A first step to achieve this goal is to determine what type of activities and contextual information can be detected from these signals using statistical learning. To this end, the way to process signals into adequate data representation, according to these detection purpose, is also a major challenge. We describe existing methods on activity monitoring in terms of type of sensor, detection tasks, features and machine learning models. On our data we use a wide feature set based on time series from various signal representations such as Fourier transform, autocorrelation and spectrograms. Using predictive models based on random forests on different experimental datasets we show Tarkett system ability to achieve various monitoring tasks, as well as the relevance of each signal representation and associated features regarding these detection tasks.

Nevertheless for these experimental studies to be deployed industrially in FIM Care real installations, machine learning models need to fulfill two crucial requirements. Firstly they have to be confronted with real environment data, meaning to be able to adapt to real installations variability and to activity signal differences between people. In this context we deal with the problem of adapting a predictive model initially trained on experimental data to real data with different empirical distribution. This particular situation in machine learning is known as *transfer learning* or *domain adaptation*. We address it by confronting simulated events data to real data on the *fall detection* task that presents the particularity of extreme *class imbalance* in real conditions. We investigate the drawbacks of this *class imbalance* on existing *transfer learning* methods on decision

trees and propose some adaptations to handle this problem. Our contribution is a robust model-based *transfer learning* algorithm on random forests able to deal with *class imbalance* and that can also be used to interpret relations between two different domains.

Secondly, most of the prediction tasks for elderly monitoring have to work in real time being embedded in an electronic device with limited computational capabilities. Taking into account this kind of constraints while designing a predictive model belongs to a branch of machine learning, known as *cost sensitive* or *budget learning*, that became an increasingly active research topic in the past years. We translate embedded system computational resource constraints into a *budgeted prediction time* framework compatible with decision tree based models and propose an efficient and scalable genetic algorithm considering both *feature acquisition cost* and *evaluation cost* allowing to pass from an experimental random forest model to a new simplified one that fits in embedded system resource limits. This algorithm takes advantage of the notion of *equivalence* between classifiers, meaning models sharing the same decision function but with different structures, to favor *feature acquisition cost* reduction by exploiting structural variety on decision trees.

Résumé (en français)

Contexte et motivations

Avec les progrès technologiques et la fiabilité grandissante des applications d'apprentissage statistique dans plusieurs domaines des sciences appliquées, les technologies basées sur des nouveaux types de capteurs et/ou des nouvelles méthodes d'apprentissage statistique se sont multipliées ces dernières années. Le projet *FloorInMotion* (FIM) de Tarkett a été développé dans ce contexte. Tarkett est une entreprise française de revêtement de sols présente dans des dizaines de pays dans le monde. Tarkett est impliqué dans de nombreux sites de construction à grande échelle comme les bureaux, les écoles ou les hôpitaux et propose un large panel de services industriels liés aux revêtement de sols. Le projet FIM représente l'ambition de Tarkett d'utiliser leurs compétences afin de contribuer au futur des applications industrielles liées aux sols dits "intelligents". En effet les capteurs sols pourraient se révéler être une technologie clef dans différents domaines tels que la sécurité, la gestion des foules, la localisation en intérieur, les sols interactifs et le monitoring dans la santé.

Le premier et principal volet de ce ce projet de sol intelligent est sanitaire et consiste à proposer des services de monitoring des personnes âgées aux EHPAD et maisons de retraite. Le système proposé est basé sur un capteur sol qui s'installe dans les chambres de ces institutions, sous le revêtement de sol, accompagné d'un boitier électronique en local pour analyser le signal en temps réel. Cela suppose de pouvoir fournir, à partir de modèles de détection appliqués sur les signaux du capteur sol, des informations précieuses sur les activités des patients âgés, ainsi que de pouvoir détecter les éventuelles anomalies et autres situations à risques telles que les chutes. Une seconde partie de l'application proposée par Tarkett consiste en un logiciel multi-plateformes pour permettre au personnel médical de suivre ces informations et d'être alertés en cas de besoin. Même sans maladies particulières les personnes âgées ont besoin d'un suivi humain important et sont souvent dépendantes sur plusieurs aspects de leur vie quotidienne. De plus, comme le personnel médical n'a pas la capacité de surveiller continuellement tous les patients, les chutes dont l'issue est fatale restent une des principales causes de mortalité non-naturelle dans ce genre d'établissements. C'est pourquoi une des grandes priorités dans le cadre de ce projet est la détection des chutes. Néanmoins, à terme l'objectif final est également d'apporter une solution technologique complète pour le monitoring des personnes âgées, regroupant plusieurs axes importants tels que : la détection de chutes, d'activités "anormales", des intrusions et sorties nocturnes, des changements d'habitudes et plus généralement la mesure des différents risques pour la santé des patients.

Un des objectifs principaux de ce travail de thèse est d'étudier expérimentalement le type d'informations qui peuvent être extraites des séries temporelles provenant du capteur piezoelectrique Tarkett, de détecter celles qui sont utiles pour le monitoring de la santé des personnes âgées et de proposer des outils d'apprentissage statistique qui sont capables de fonctionner dans les environnements réels où les ressources

computationnelles sont limitées.

Monitoring de l'activité quotidienne des personnes âgées

En France chaque année environ 10000 morts (20000 aux Etats-Unis [87]) et des milliers de blessures très sévères de personnes âgées sont dues aux chutes et les conséquences de celles-ci pourraient être largement limitées si ces accidents étaient détectés assez tôt [231]. De surcroît, plusieurs études médicales montrent que le risque de chute peut être anticipé par l'analyse des capacités motrices du patient, notamment au niveau de la marche [37, 47, 84, 116, 208].

En environnement médical l'apprentissage statistique peut avoir une plus-value à plusieurs niveaux [202], en fonction de la manière dont il aide les professionnels de santé dans leur travail. Quelques une de ses applications sont utilisées dans les opérations médicales comme les chirurgies, d'autres sont utiles pour aider à établir des diagnostics [88], et certaines offrent des outils aux professionnels pour mieux surveiller l'état de santé de leurs patients et pour les alerter en cas de risque pour la santé du patient [28, 83]. Le suivi de la santé à travers le monitoring des activités quotidiennes appartient à cette dernière catégorie d'applications. Il présente un avantage notable chez les personnes fragiles telles que les personnes âgées et peut permettre l'analyse globale des habitudes de vie pour détecter les changements anormaux ou les accidents. La majorité de ces tâches sont evidemment à la portée des personnels de santé mais le but de ce genre d'applications automatisées est surtout de leur faciliter la tâche de plusieurs manières dans la gestion de leurs patients. Premièrement, ce genre d'applications peuvent palier au manque de personnel dans certaines situations (typiquement le cas pour la détection de chute). En outre, elles peuvent également servir à compléter les informations auxquelles ont accès les médecins pour leur permettre de traîter plus efficacement leurs patients.

D'une manière plus générale, l'apprentissage statistique pourrait aussi être utile pour organiser plus efficacement les différents efforts de santé, en terme de ressources humaines par exemple. En effet, plusieurs pays ont récemment expérimenté, au travers de la crise du covid-19, que même si le volume du personnel de santé peut être considéré comme suffisant dans des conditions normales, cela peut changer très rapidement en situations extrêmes. Nous observons que malgré les connaissances très récentes sur cette maladie, elle est déjà largement sujette à des recherches en apprentissage statistique, que ce soit pour la détection ou le diagnostic détaillé [145]. Ainsi, les tâches de routine du monitoring pour la santé qui sont à la portée de l'apprentissage statistique sont précieuses pour le corps médical et peuvent représenter un gain dans la qualité des soins. C'est la raison d'être du développement du système *FIM Care* de Tarkett pour les établissements de santé.

Quelles informations extraire de l'activité humaine ?

Le monitoring de l'activité humaine est un sujet sensible étant donné qu'il est directement relié à la vie privée des personnes. En ce qui concerne le monitoring de l'activité quotidienne des patients il est nécessaire d'une part d'extraire assez d'informations utiles pour en avoir une application médicale et d'autre part que ces informations ne portent pas atteinte à la vie privée des patients. Pour cette raison des capteurs trop

intrusifs tels que les caméras ou les microphones sont difficilement viables pour le monitoring de la vie quotidienne malgré la précision des informations qu'ils captent. Quant aux capteurs qui se portent sur le corps, ils nécessitent que les patients les utilisent correctement et systématiquement dans leur vie quotidienne, ce qui est difficilement fiable pour les personnes âgées. Le choix de Tarkett pour les capteurs au sol apparaît alors particulièrement adapté en terme de non-intrusivité, de fiabilité et de facilité d'usage. Il est important de préciser que chaque type de capteur possède des propriétés du signal particulières qui déterminent l'information qui peut en être extraite et, par conséquent, le périmètre des applications qui peuvent l'exploiter. Par exemple, pour ce qui est des capteurs au sol, le poids d'une personne est facilement mesurable avec un capteur de pression statique mais beaucoup plus délicat avec un capteur de pression dynamique. Il peut théoriquement être approximativement déductible avec une caméra, moyennant une modélisation relativement fine, mais cela paraît difficilement réalisable avec des microphones (bien que ce ne soit pas nécessairement impossible).

Malgré les avantages évoqués des capteurs au sol pour le monitoring de l'activité quotidienne, les capteurs couvrant une large surface peuvent difficilement apporter autant d'information que des caméras et ont une qualité du signal moindre comparés aux capteurs portables. C'est pourquoi la problématique de l'obtention d'une représentation des données utile et interprétable à partir de séries temporelles unidimensionnelles est importante, tout comme la conception des features en fonction des tâches de monitoring visées. Le dernier niveau d'information concernant les activités monitorées est lié aux modèles d'apprentissage statistique qui sont entraînés à partir des données capteur et ce choix des modèles conditionne autant leur efficacité que leur interprétabilité.

Adaptation à différents environnements et technologies

Que ce soit pour des applications médicales ou pour d'autres applications industrielles, les modèles d'apprentissage statistiques sont souvent confrontés aux mêmes problèmes pratiques. En général les bases de données utilisées pour entraîner statistiquement ces modèles sont supposées refléter de manière assez proche la distribution des données réelles, cependant c'est rarement le cas en pratique en raison des biais divers qui dépendent du processus de récolte des données. Ce processus depend de différents facteurs qui sont plus ou moins contrôlables en fonction de l'application mais une situation fréquente et générale est de devoir manier plusieurs sources de données provenant de différents environnements. Une autre source d'hétérogénéité des données provient de la variabilité de la technologie elle-même. En effet en conditions réelles les capteurs sont souvent soumis à des défauts, soit dans le conception technique soit dans la manière dont ils sont utilisés. De plus, la plupart des applications industrielles innovantes visent à mettre à jour leur technologie rapidement lorsqu'elles identifient de possibles améliorations (par exemple en terme de qualité du signal, de ressources computationnelles ou encore d'aspects économiques), ce qui peut également être source d'hétérogénéité.

Exploiter des domaines de données différents bien qu'apparentés en tirant parti de cette hétérogénéité est le principe du *transfer learning* [191, 260], et prendre en compte les contraintes sur les ressources technologiques dans l'apprentissage statistique est le principe du *budgeted learning* [93]. Ces deux thématiques de recherche sont d'une importance capitale en ce qui concerne les applications industrielles à grande échelle et

le système de monitoring des personnes âgées proposé par Tarkett est une illustration typique de leur intérêt.

Organisation du manuscrit

- Chapitre 1 : Système de monitoring basé sur un capteur sol piezoelectrique

  Ce premier chapitre compare le système FIM Care avec d'autres technologies de monitoring de l'activité humaine et détaille ses spécificités en terme de technologie, de données et de conditions requises pour pouvoir être installé à grande échelle dans les établissements de santé. L'installation complète du capteur dans les chambres des patients est décrite et la physique du signal piezoelectrique est détaillée, ainsi que le circuit électronique nécessaire pour amplifier ce signal et ses conséquences. Ceci permet de mettre en évidence certaines propriétés du signal afin de mieux cerner le type d'applications qui peuvent être visées par la technologie Tarkett. En effet d'autres applications utilisant le même capteur (le film piezoelectrique Emfit) [132, 189, 197] sont présentées et nous expliquons en quoi le système Tarkett FIM Care se distingue de ces applications et pourquoi il implique de tout nouveaux challenges techniques. A la fin du chapitre sont présentés une vue globale du système complet dans les différents environnements où il est installé, le procédé de récolte des données et une description des différentes bases de données disponibles pour l'apprentissage statistique.

- Chapitre 2 : Représentation des séries temporelles et modèles prédictifs d'arbres de décision

  Ce chapitre présente tout d'abord plusieurs représentations des séries temporelles, comme l'autocorrelation et les spectrogrammes, pour concevoir un espace de features efficace et flexible qui peut être employé pour différentes tâches de classification de l'activité humaine basées sur les signaux de pression au sol. Ces représentations des séries temporelles et les features correspondants montrent leur utilité sur plusieurs types de données, où des modèles fôrets aléatoires présentent de bonnes performances de classification, notamment pour la *détection de chute*, la détection de *mouvements au sol* et la reconnaissance de la *marche* des personnes âgées. Chaque modèle est entraîné à partir d'un jeu de données labélisées représentant différents types d'évènements qui sont récoltés à partir de trois installations expérimentales différentes. En fonction du type d'évènement à détecter et du feedback sur ces modèles expérimentaux, chacun est associé à une stratégie d'implémentation particulière, comme le lissage de la réponse instantanée des fôrets aléatoires pour réduire le taux de fausses alarmes, l'utilisation de classifieurs en cascade pour augmenter l'interprétabilité et la performance dans le cas de la classification multi-label, ou encore la déduction directe de certains paramètres physiologiques comme la vitesse de marche à partir de features particuliers. Les résultats présentés dans ce chapitre montrent que Tarkett FIM Care est viable en tant que système complet de monitoring de l'activité quotidienne des personnes âgées pour des applications médicales.

  Cependant pour exporter la plupart de ces modèles expérimentaux de prédiction en conditions réelles, ils doivent pouvoir tourner en temps réel dans un

système embarqué. Ainsi dans le contexte de la prédiction en temps réel avec des ressources limitées, un des principaux obstacles est la taille, la complexité et la redondance de l'espace de features. Pour cette raison, ce chapitre étudie également la pertinence des différentes représentations des séries temporelles selon les différentes tâches de détection, ainsi que la question de la redondance des features selon trois métriques : l'importance des features relative aux fôrets aléatoires, les corrélations statistiques et le mRMR (minimum redundancy maximum relevance). Ce dernier critère suggère que s'il est possible de détecter les corrélations entre les features même dans le cas non-labélisé, la pertinence d'un sous-ensemble de features dépend intimement des tâches détection. Néanmoins, toutes ces métriques ne prennent pas en compte la diversité des features en terme de complexité computationnelle et c'est la raison pour laquelle cette problématique spécifique est explorée en détails dans le chapitre 4.

- Chapitre 3 : Transfer learning : passage des situations expérimentales à l'environnement opérationnel

  La première implémentation réelle du modèle de détection de chute dans des chambres de patient, détaillé dans [175], a été entraînée sur des données expérimentales de chute décrites dans le chapitre précédent. Après un an de récolte de données réelles à partir des feedbacks du modèle initial, un nouveau jeu de données sur la détection de chute a été constitué et confronté dans ce chapitre avec le premier jeu de données expérimental, pour souligner la problématique de l'adaptation du modèle des données expérimentales vers les données réelles. Sans surprise, cette comparaison montre que les distributions empiriques des données diffèrent entre ces deux bases de données. Une des différences les plus directement observables étant l'important *déséquilibre de classes* des données réelles. En effet, en réalité les chutes restent très rares comparées aux autres évènements de la vie quotidienne, contrairement à la base de données expérimentale où les chutes et non-chutes sont représentées à peu près équitablement. Cette dissimilarité des jeux de données souligne le besoin d'adapter le modèle initial de détection de chute, qui sert de référence, sans le dégrader. Dans cette optique nous examinons les possibilités de *transfer learning* agissant directement sur les modèles et aboutissons à plusieurs adaptations de deux algorithmes de transfert existants sur les arbres de décision pour la situation de déséquilibre de classes, en particulier au travers des notions de *risque de pruning* et de *déséquilibre homogène des classes*. En utilisant ces adaptations nous proposons un nouvel algorithme de transfert général sur les fôrets aléatoires qui sélectionne la meilleure version transférée pour chaque arbre sur des échantillons bootstrap. Cela résulte en une fôret aux arbres complémentaires capable d'une part de gérer le *déséquilibre de classes* mais aussi d'apporter de précieuses indications sur le type de relations entre les deux jeux de données.

- Chapitre 4 : Budgeted learning : mise à l'échelle industrielle de modèles d'arbres de décision pour les systèmes embarqués

  Le dernier chapitre aborde le problème des modèles d'arbres de décision en situation de ressources limitées. Il se concentre sur les ressources computationnelles des systèmes embarqués et sur la contrainte qu'elles impliquent sur le temps de

prédiction des modèles. Ce problème est plus connu sous le nom de *temps de prédiction budgeté*. En prenant en compte à la fois les coûts *d'acquisition des features* et les coûts *d'évaluation du modèle* sur les ensembles d'arbres de décision, nous formalisons de manière générale le *temps de prédiction* des modèles d'arbres de décision et nous examinons le problème associé d'optimisation sous contraintes. Plusieurs cas particuliers correspondants à des travaux existants découlent de ces définitions. Pour quantifier précisément les coûts computationnels liés à notre application, nous introduisons également la situation de coûts de calculs partagés par des groupes de features, ce qui correspond dans notre cas au calcul des représentations des séries temporelles. Après avoir mis en évidence que les coûts d'acquisition des features sont intimement liés à la structure des arbres de décision et l'ordre d'utilisation des features au sein de ces arbres, nous définissons la notion *d'équivalence* entre les arbres de décision, faisant référence aux arbres qui partagent la même fonction de décision mais pas la même structure. Cette notion est alors exploitée dans un algorithme génétique de pruning aléatoire pour résoudre efficacement le problème d'apprentissage budgété auquel nous sommes confrontés, pour pouvoir obtenir finalement une fôret aléatoire qui respecte les contraintes de ressources d'un système embarqué. Cet algorithme génétique est très flexible dans la mesure où il est compatible avec n'importe quelle définition des coûts, en particulier pour notre situation de coûts partagés, et peut théoriquement prendre en compte plusieurs contraintes à la fois. De plus, les expériences faites sur les données synthétiques et les données de chutes suggèrent que la notion *d'équivalence* entre les modèles de classification peut être exploitée de façon pertinente pour ce genre de problèmes.

Publications

- L. Minvielle, M. Atiq, R. Serra, M. Mougeot and N. Vayatis. Fall detection using smart floor sensor and supervised learning. *39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), pages 3445-3448, 2017.* https://ieeexplore.ieee.org/document/8037597

- L. Minvielle, M. Atiq, S. Peignier and M. Mougeot. Transfer Learning on Decision Tree with Class Imbalance. *IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI),2019, pages 1003-1010.* https://ieeexplore.ieee.org/document/8995296

- M. Atiq, S. Peignier, M. Mougeot. Constrained prediction time random forests using equivalent trees and genetic programming : application to fall detection model embedding. *IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI), 2021.* https://ieeexplore.ieee.org/document/9643373

- M. Atiq, L. Minvielle, C. Truong, R. Serra, M. Mougeot, N. Vayatis. A Data Set for Fall Detection with Smart Floor Sensors. 2021. (submitted)

Introduction

Context and motivations

Because of technological progresses and reliability of machine learning applications in numerous applied science fields, technologies based on new sensors and/or new machine learning methods exploded in the last recent years. The *FloorInMotion* (FIM) project of Tarkett was developed in this context. Tarkett is a french flooring company present in dozens of countries. Tarkett is involved in large scale construction sites such as offices, schools or hospitals and proposes a large variety of floor related services. The FIM project represents their ambition to use their skills to contribute to future large areas smart floor applications. Indeed this kind of large scale floor sensors could be a key technology for innovative applications in various fields such as security, people flow management, indoor localization, interactive flooring or daily living monitoring in the near future.

The first and main objective of this smart flooring project, namely *FIM Care*, is to build an elderly monitoring solution dedicated to EHPAD and nursing home clients. The proposed application consists in installing in every room directly under the flooring a piezo-electric sensor with a local electronic hardware to analyze the recorded sensor signal in real time. This implies to provide helpful information about elderly patient activities based on predictive models applied on flooring signal and being able to detect anomalies and risky situations such as falls. The second part of this application is a remote multi-platforms software for the medical staff to be able to follow these A.I. monitoring insights and be alerted instantly if needed. Even without particular disease, elderly patients usually need important human resources to take care of them as they are often dependent in their daily life activities. Moreover, as medical staff can not watch carefully 24h/24 every patient, fatal falls for elderly people are still a major non-natural death cause in nursing homes. So one of Tarkett's priorities with *FIM Care* is the *fall detection*. However the final purpose is also to bring a complete technological solution for elderly monitoring, gathering several important functionalities like : fall detection, abnormal activity detection, intrusion and night leaves detection, habit changes detection, health risk assessment.

One of the main purposes of this work is firstly to study experimentally what kind of information can be extracted from Tarkett piezo-electric smart-floor time series that are helpful for elderly daily activities monitoring and to provide efficient tools to make machine learning models work in real environments with limited computational resources.

Monitoring daily life monitoring of elderly people

Concerning elderly people, in France around 10000 deaths each year (20000 in the U.S. [87]) and a lot more severe injuries are due to falls and these consequences could be importantly lowered if accidents were detected rapidly enough [231]. Moreover medical

researches show that the fall risk can be anticipated by analyzing patient natural motion, especially the gait [37, 47, 84, 116, 208].

In a medical environment machine learning can bring added value at several levels [202], depending on how it helps medical professionals in their work. Some applications are used in medical interventions like surgery, others are helpful in establishing diagnosis [88] and some of them offer tools for medical professionals to monitor patients health state and to alert them if any risk for a patient health is detected [28, 83]. Daily life health monitoring belongs to this last category of application. It may help to monitor the most fragile people like elderly, using routine activities analysis to detect any abnormal change or accident. The majority of these tasks can be done successfully by professionals but the purpose of these kinds of machine learning applications is to help them to manage patient health in several ways. First it can compensate for the potential lack of needed human resources in certain situations, secondly, it can be used to bring important insights to experts making them treat their patient faster and in a more adapted way.

In a general sense, machine learning may be useful for organizing more efficiently professional healthcare human resources efforts. Indeed, a lot of countries recently experimented with covid-19 that even if the amount of human resources seems sufficient to monitor patients health in usual conditions, extreme situations show that it can change rapidly at any moment. Despite the newness of the knowledge about this disease, it is already subject to machine learning researches in terms of automated detection [145]. Thus, routine monitoring tasks that can be in the scope of machine learning are precious for medical staff and represent an important potential improvement in healthcare monitoring. So that is what *FIM Care* system in elderly healthcare institution is made for.

What information to collect from human activity ?

Human activity monitoring is a sensitive topic as it is directly linked with people privacy. Concerning elderly daily life monitoring it is necessary to extract enough useful information about people activity for healthcare application without any information that could directly break privacy. For this reason some intrusive sensors like microphones or cameras are hardly viable although their ability to provide very accurate information. Wearable sensors require patients to use them properly and consistently in their daily life, which is not reliable enough for elderly. The choice of Tarkett for a flooring pressure sensor seems particularly appropriate in terms of non-intrusiveness, ease of use and reliability. It should be mentioned that each kind of sensor has its own signal properties which determines information that can be extracted from it and therefore the range of applications that can be built upon it. For example a person's weight is easily deductible with a static pressure sensor whereas it is a lot harder with a dynamic pressure sensor. It could be approximately estimated used cameras with a suited sophisticated model and this would appear very difficult using only microphones (but not necessarily impossible).

Despite their advantages for daily life monitoring large floor sensors can hardly provide as much information as cameras and as much signal quality as wearables. Then the topic of obtaining a good and interpretable data representation of 1-dimensional time series becomes important and so is feature designing relatively to targeted monitoring

tasks. The final layer of information extracted human activities comes from machine learning models that are built using sensors data and their choice determines as much monitoring tasks efficiency as comprehensibility in terms of response interpretation and richness of feedbacks.

Adaptation to different environments and technologies

Whether it is for medical purpose or other industrial applications machine learning models are often confronted with the same practical problems. Usually databases used to statistically train them are supposed to reflect closely real data distribution, but it is rarely the case in practice because of numerous biases relying on the data gathering process. This process depends on various variables that are more or less controllable depending on the application, but a frequently encountered situation is to deal with several sources of data corresponding to different environments. Another cause of data heterogeneity is the variability of the technology itself. Indeed in real conditions sensors are often subject to some defaults either in their technological design or in the way they are used by humans. Moreover most of innovative applications aim at updating their technology as soon as possible if it can provide any form of enhancement (for example in terms of signal quality, computational resources or even economic aspects) and this can also be a source of data heterogeneity.

Dealing with data domains that are different although related by turning this heterogeneity into an advantage is the purpose of *transfer learning* [191, 260] and taking into account technological resource constraints in statistical learning is known as *budgeted learning* [93]. Both of these research fields are particularly important while considering large scale industrial applications of machine learning and Tarkett elderly monitoring system is a typical illustration of their relevance.

Organization of the manuscript

- Chapter 1 : Piezoelectric smartfloor monitoring system

  This first chapter of the manuscript compares Tarkett FIM Care system with other kinds of human activity monitoring systems and precise its particularities and tools to develop a reliable monitoring system in terms of technology, data and requirements to be implemented in large scale in elderly health-care institutions. Patient room flooring installation is described and we detail physics of the piezo-electric signal coming from the sensor as well as the electronic circuit needed to amplify it and its consequences on the signal. This allows to point out some properties of the signal to better understand what type of applications can be targeted based on Tarkett technology. In particular some alternative applications of the same sensor (Emfit piezo-electric film)[132, 189, 197] are mentioned and we explain why Tarkett FIM Care system very distinguishable from these applications and the new technical challenges it implies. At the end of chapter, we present an overview of the whole system from the different environments where this smart-floor is installed, to the data gathering process and we describe various databases available for machine learning use.

- Chapter 2 : Time series representation and decision tree predictive models

This chapter proposes various time series representations like spectrograms and autocorrelation for designing an efficient and flexible feature set that can be used for various human activity classification tasks based on floor pressure signals. These time series representations and the corresponding features show their suitability on several experimental and real data, using random forest models with good classification performance on *fall detection*, *lying on-floor activity detection* and *elderly walk recognition*. Each model is associated with a labeled dataset representing various kinds of events and recorded with 3 different experimental installations. Based on the type of event to detect and the feedbacks of these experimental models, each one has a specific implementation strategies, such as smoothing random forest real-time outputs to reduce false alarm rate, using cascade classifier to increase interpretability and performance with multi-label classification or deducing from particular features some physical parameter assessment like gait speed. Results presented in this chapter show that Tarkett FIM Care is viable as a complete elderly daily life activity monitoring for health-care purposes.

Nevertheless, to export most of these experimental predictive models into real conditions, they must be run almost in real time in an embedded system. So the computations needed to process features are also importantly limited. Therefore, in this context of real time prediction with limited resources, the main obstacle relies on size, complexity and redundancy of the feature set. Thus this chapter also studies the relevance of each time series representation according to the different detection tasks as well as feature redundancy problem through three main metrics : random forests feature importance, statistical correlation and mRMR (minimum redundancy maximum relevance). This suggests that if it is possible to detect correlations between certain features even in unlabeled situations, the relevance of feature subsets depends a lot on the detection tasks. However these metrics ignore the variability in features computational complexity and that is the reason why this specific problem is explored in details in Chapter 4.

- Chapter 3 : Transfer learning : from experimental setup to operational environment

  The first fall detection model implementation in real patient rooms, detailed in [175], has been trained on the experimental fall data described in the previous chapter. After one year gathering real data based on feedbacks of this initial model, a new fall detection dataset has been obtained and is confronted in this chapter with the primary experimental dataset, in order to point out the problem of model adaptation from experimental data to real ones. This comparison shows non-surprisingly that empirical data distributions are different between these two datasets. One of the most straightforward observable difference being the hard *class imbalance* of real data. Indeed falls are very rare in proportion in reality compared to the experimental dataset where fall and non-fall event are broadly equally represented. This data dissimilarity motivates the need of adapting the previous fall detection model that serves as a reference without degrading it. For this purpose we examine model-based *transfer learning* possibilities and come up with several adaptations of two existing transfer algorithms on decision trees for class imbalance situation, especially by using notions of *pruning risk* and

homogeneous class imbalance. Using these adaptations we propose a new general transfer algorithm on random forest that selects a best transferred variant on bootstrapped samples for each tree, resulting in a complementary random forest able to handle *class imbalance* but also to bring valuable clues about the type of relations between the two different data distributions.

- Chapter 4 : Budgeted learning : industrial scaling up of decision tree models for embedded systems

  The last chapter addresses the problem of resource constrained decision tree based models. It focuses on computational resources of embedded systems and how it constraints machine learning models *prediction time*. This problem is also known as the *budgeted prediction time* problem. Taking into account both *feature acquisition cost* and *evaluation cost*, we define a general formulation of the *prediction time cost* of decision tree based models and we examine the associated constrained optimization problem. Several particular cases corresponding to existing works can be derived from this definition. To quantify precisely the computational costs related to our application, we also introduce the possibility of shared computation costs for groups of features that correspond in our case to time series representations computation. After pointing out that decision trees feature acquisition costs are tightly linked with their structure and the order of feature usage we define the notion of *equivalence* between decision trees, referring to trees that share the same decision function but not the same structure. Then this notion is used in a genetic random pruning algorithm to solve efficiently the *budgeted learning* problem we are facing to be able to get a random forest model fitting embedded system resource constraints. This genetic algorithm is very flexible as it is compatible with any computation cost definition, in particular our shared computation situation, and can also theoretically handle several budgets at the same time. Moreover through experiments made on synthetic data and fall data we show that *equivalence* between classification models can be relevant to exploit for this kind of problems.

Publications

- L. Minvielle, M. Atiq, R. Serra, M. Mougeot and N. Vayatis. Fall detection using smart floor sensor and supervised learning. *39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), pages 3445-3448, 2017*. https://ieeexplore.ieee.org/document/8037597

- L. Minvielle, M. Atiq, S. Peignier and M. Mougeot. Transfer Learning on Decision Tree with Class Imbalance. *IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI),2019, pages 1003-1010*. https://ieeexplore.ieee.org/document/8995296

- M. Atiq, S. Peignier, M. Mougeot. Constrained prediction time random forests using equivalent trees and genetic programming : application to fall detection model embedding. *IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI), 2021*. https://ieeexplore.ieee.org/document/9643373

- L. Minvielle, M. Atiq, C. Truong, R. Serra, M. Mougeot, N. Vayatis. A Data Set for Fall Detection with Smart Floor Sensors. 2021. (submitted)

Notations

- ADC : Analog-digital converter
- AUC : Area under curve
- BudGenPrune : genetic pruning algorithm for budgeted learning
- DB : Database
- DG : Divergence gain
- DT : Decision tree
- FPR/TPR : False positive rate / True positive rate
- HAR : Human activity recognition
- IG : Information gain
- KL : Kullback-Leibler divergence
- JSD : Jensen-Shannon divergence
- MAB : Multi-armed bandit
- MI : Mutual information
- mRMR : Minimum redundancy maximum relevance
- OOB : Out-of-bag
- PRR : Pruning risk
- ROC : Receiver operating characteristic curve (FPR/TPR curve)
- SER : Structure Expansion/Reduction algorithm
- SER_{NP} : SER "No Prune" algorithm (variant with pruning restriction)
- STRF : Selective Transferred Random Forest algorithm
- $STRUT$: Structure Transfer algorithm
- $STRUT_{ND}$: $STRUT$ "No Div." algorithm (variant without using DG)
- $STRUT_{NP}$: $STRUT$ "No Prune" algorithm (variant with pruning restriction)
- RF : random forest
- TS : time series

List of Figures

List of Tables

1

Piezoelectric smartfloor monitoring system

1 Real time health monitoring

Miniaturization and decreasing costs of physical sensors, as well as the growth of IoT use and progresses in signal processing and machine learning, extended the range of applications and concerned fields using these technologies in recent years. It is even observable for general public through smartphones and smartwatches that embed more and more of these physical sensor-based applications such as sport or sleep monitoring. Using only usual smartphone sensors (as gyroscope, accelerometer and magnetometer), it is possible to perform walk detection and step counting with relatively simple algorithms (threshold based on signal standard deviation and energy values) and quite accurately (less than 3% of error) [33]. Moreover cameras are now very affordable and progresses in computer vision allowed the raise of visual security and surveillance [140]. Nowadays a large variety of sensors and methods are developed for various human activity monitoring applications. In the mean time of these technological advancements, the proportion of elderly people in "western" societies is increasing implying daily life dependency concerns. Furthermore, health diagnosis and interventions are more and more relying on complex data analysis and machine learning. In addition, for countries with already advanced healthcare systems like France, a new concern is preventive medicine of fragile people.

In the following, existing monitoring systems are presented, organized by the kind of sensors they use, advantages and drawbacks of these different types of sensor and a detection task oriented overview of health monitoring systems.

1.1 Sensors

Automatic health monitoring is a wide topic involving several challenges as indoor tracking, activity recognition, physical or physiological activity analysis and anomaly detection. There exist numerous systems used for these purposes and one natural way to categorize them is by the type of sensor they rely on. A detailed overview of existing technologies, how they work in terms of physics and for which tasks they are employed is proposed in [162]. All these monitoring systems can be divided into four main groups : vision or acoustic based sensors, wearable sensors, ambient sensors and hybrid systems.

Vision and sound based sensors

Cameras and microphones based systems are the closest to human perceptions. A lot of daily life activities (cooking, bathing, walking, etc..) and anomalies (falls, unusual night activity, etc..) can be easily recognized by vision or sound by a human. Vision based systems provide a quite precise information about patients and their environment, especially concerning spatial information. From more than a decade, there are efficient existing methods for human or object recognition and tracking, for unique cameras in small indoor areas as well as extended areas monitored by several cameras [140]. For example the application presented in [186] also shows that basic tracking can be sufficient to detect some activities in an office-like environment. In the same way, some daily life activities like making coffee, sleeping or taking a shower are guessable only by measuring the time spent on some precise locations. Daily activity monitoring systems described in [82] and [81] use several cameras with a tracking module in a

living environment to segment these locations and duration sequences for extracting daily activities, patient habits and even abnormalities regarding these habits.

There are numerous advantages of vision based monitoring systems : first the literature richness on existing detection methods [140] and datasets [51] represents a good starting point for designing a new vision based monitoring system; secondly this kind of systems can provide an accurate spatial information, more particularly with depth cameras [278], which allows patient tracking, object interaction detection and simplifies fall detection [127, 281]. However vision based systems are not very robust to variability and evolution of living environment where they are deployed. Indeed all kinds of cameras are sensitive to the occlusion problem, which is only avoidable using several cameras, and need a new calibration phase each time they are confronted to a new living environment. But the main drawback of vision based monitoring systems is unavoidable as it is their intrusiveness.

Another type of monitoring systems that is less sensitive to environment changes and occlusion problems is the family of acoustic based sensors, while being still relatively human interpretable. For instance, a bathroom activity monitoring system based on a unique microphone is proposed in [53]. It is designed to provide a daily report for caregivers with five different activities (showering, urination, flushing, washing hands and sighing) and reaches a total accuracy of 87%. Acoustic based fall detection have also been investigated in the case of a unique microphone [283] or using several ones [154]. Microphones are usually cheaper than cameras, moreover they suffer less from the occlusion problem and need less calibration effort. Nevertheless, using several of them can be necessary to reach best performances, they are still quite sensible to environment perturbations (noises, signal degradation due to obstacles) and they do not solve the intrusiveness issue.

Wearable sensors

Wearable sensors are another highly employed kind of sensors for health monitoring. As the majority of them does not record any image or sound, they might produce less surveillance/intrusiveness feeling compared to vision or acoustic based sensors but are also less convenient for patients if they have to wear these sensors permanently. Wearable sensors are largely used by doctors for diagnosis purposes in limited time situations, especially for medical consultations. For instance, [180] uses several kinematic and electromyographic wearable sensors for gait patterns classification. Authors followed dozens of stroke recovering patients at different phases in order to extract relevant variables to discriminate, through gait analysis, different clusters corresponding to medical patient states. As they provide diversified and very precise information about motions, wearable sensors are crucial to build bridges between medical knowledge and automatic sensor monitoring and can allow highly efficient detection rates for precise body motion recognition like gait analysis [180, 274] or fall detection [128, 152].

A review of recent monitoring systems based on wearable sensors is proposed in [179]. It presents several monitoring technology architectures, distinguishes them based on used methodologies and the monitored activities. It also gives general insights about what kind of technical difficulties one may have to face while designing a wearable monitoring device. Another recent survey provides a more detailed categorization of existing works on wearable sensors monitoring in function of sensors type, the exact

part of the body where they are carried and the precise activity they are able to detect [61] .

Most common used wearables for activity classification are inertial sensors like accelerometers, gyroscopes and magnetometers. Indeed 3-axial accelerometers provides a precise information about the body motion because, as long as the sensor is well-fixed on a body part, it is possible to deduce very accurately how this body part is moving. Gyroscopes and magnetometers are mainly used as supporting sensors in combination with accelerometers to correct inaccuracies of the latter, especially when it is not constrained to a fixed position [222]. Nevertheless, existing works show that some activity recognition is possible while using gyroscopes or even magnetometers alone. For example, [29] presents an intuitive threshold-based fall detection algorithm using bi-axial gyroscope features, [148] shows that gyroscopes for activity recognition are replaceable, describing an approach for estimating angular velocity with magnetometers, and [7] uses a unique magnetometer for measuring walking and running cadence. These kinds of inertial sensors are now findable in every smartphone which makes it an usable device for human activity recognition [158].

Ambient sensors

Ambient sensors regroup all other sensors sensitive to the monitoring environment without being as intrusive or uncomfortable as cameras, microphones or wearables. The principle of ambient sensors is to cover the whole patient living area with sensors that are not a direct issue for patient privacy. A very recent review of this kind of sensors under the name of "device-free" systems for human activity recognition presents and compares recent systems (the last ten years) on technology, application and algorithmic aspects [123].

One particular approach to monitor a whole living area is by covering it with a floor sensor. Existing works with these kinds of sensor show their efficiency in people tracking [120], gait recognition [171, 218], fall detection [209] and even people identification [230, 241], which makes the use of floor sensors a reliable and unobstrusive way to monitor elderly people. Floor sensor based systems can be classified considering two main criteria impacting methods that can be used and achievable applications. On one hand, depending on the floor sensor technology, pressure forces are perceived either in a static way or in a dynamic way. More precisely, floor sensors are either sensible to the amount of pressure forces applied or to the variation of these forces. Usually capacitive sensors (using surface electric field) use capacitance or impedance changes induced by the floor deformation [12] or the human body itself [120] and are therefore able to detect static pressure forces. Conversely, piezoelectric sensors made from electret materials (usually charged polymers) [189], like the one used by Tarkett, work like a "sponge" filled with electric charges which generates charge signals only at the moment of floor pressure variation, like impacts, and are then more adapted to dynamic events detection. On the other hand, another way to distinguish floor sensor based systems is whether they rely more on floor spatial information or pressure force values. So systems with high spatial resolution often use a matrix of pressure sensing units and base their detection on shapes of contact surfaces on the ground [12, 171, 219], whereas the others usually use one accurate signal to detect events [11, 218, 236].

Another kind of ambient sensor systems uses the electromagnetic field and its

variation to perceive moving objects and people in the environment with numerous physics approaches. This is the case for Wifi based systems, radars and other radio frequency related sensors. As Wifi systems are already deployed for communication in almost every human indoor environment and with their relative non-obstructive property, they are good candidates for low-cost people motion related detection tasks. For this reason numerous applications use Wifi signals for people motion monitoring purposes. As it is explained in a recent survey on Wifi based smart home systems [129], Wifi based people detection applications can be classified in four categories : people identification, contextual information acquisition like tracking, gesture recognition and health monitoring. These applications rely on two main indicators : the received signal strength indicator (RSSI) and the channel state information indicator (CSI), which are two measures impacted by the presence of obstacles between emitter and receiver devices. With this approach, existing works show that it is possible to estimate the speed of body parts motion [249] and then to perform gait recognition [250].

Wifi is a particular communication protocol used within a particular range of radio frequencies but the same approach can be applied with other kinds of radio signals. Indeed several existing works show the feasibility of human tracking by body reflection of radio signals. Using these methods, [121] proposes precise gait features assessment, like velocity and stride length, with a unique radio emitter hung to a wall. Another common radio frequency technique for people motion detection is the spectrogram-based approach with radar and the Doppler effect. The Doppler effect is while a moving object induces frequency shifts on electromagnetic waves. Considering human bodies, all the different articulated parts of the body that are moving induce a small Doppler effect, known as micro-Doppler signatures. Thus some existing works show that accurate motion recognition can be done, whether it is applied for gait classification [232] or more extended activity recognition [86], by studying time-frequency representations of these micro-Doppler signatures.

Binary output sensors and hybrid sensor systems

Binary output sensors can be made of plenty of different technologies but they all have the same purpose : to indicate a state change from interactions between human and objects in an actuator way. For example they can be used to detect doors opening and closing, lights activation, shower and toilets activation, kitchen devices usage, presence on bed or chairs, etc... They can represent a relatively simple way to achieve daily life activity recognition [239, 243, 244] through object interactions and allow also to recognize habits and then life behavioral changes and anomalies [213, 229]. Nevertheless, for deploying these kinds of system, the living environment, usually composed by several rooms with these monitoring systems, has to be full of daily life objects to interact with [170] and it is not always the case for every nursing home. Indeed the richness of information obtained with binary sensor based systems and then their efficiency in term of monitoring depends on the amount of used sensors and the frequency of their usage. Finally unlike ambient monitoring systems, binary sensor based systems often let a lot of undetectable areas, making them not sufficient for fall detection, gait recognition or any motion related detection.

For this reason, a common approach is to combine binary sensors with another kind of sensor like cameras, microphones or floor sensors [81, 82, 168, 169]. This idea

of designing a monitoring system out of several complementary sensors is called *hybrid systems* and allows to compensate lacks of one sensor or just to confirm detection in some situations. For instance to achieve fall detection the system in [233] uses a combination of passive infrared sensors, microphones and vibration sensors, whereas the one presented in [236] uses infrared camera activation to confirm a first detection done by a floor sensor.

Summary of each sensor particularities

The nature of information that is returned from all of these sensors is important to determine which kind of application can be based on. Indeed it would be hardly achievable to estimate physiological parameters like heart beat with just a distant camera or to track accurately a patient with only one microphone. Thus being aware of advantages and weaknesses of each sensor's type can help to select a suitable technology relatively to a given health monitoring application. So the different kinds of sensor presented previously are organized in Table 1.1 according to several criteria that are important to consider while designing any health monitoring system.

Sensor	Signal type	Interpr.	Spatial info.	Non-intrus.	Non-interf.
Camera	Image	●●●●	●●●●	○○○○	●○○○
Microphone	Sound	●●●○	●○○○	○○○○	●●○○
Wearable	Kin./physio.	●●●○	●●●○	●○○○	●●●●
Radar/wifi	EM field	●○○○	●●○○	●●●○	●●○○
Floor	Pressure F.	●●○○	●●○○	●●●●	●●●○
Binary	Object interac.	●●●●	●●●○	●●○○	●●●●

Table 1.1: Comparison of health monitoring system sensors

The first criterion in this comparison is the *non-intrusiveness* of the monitoring system. Indeed patient users to monitor tend to accept more certain devices than others, especially elderly, for privacy or comfort reasons and it is a crucial aspect for daily life monitoring. The second one is *human interpretability* of data which is important to anticipate for many long-term monitoring applications that aim at enhancing progressively their technology. It impacts the ease of labeling process, miss-detection or other monitoring errors analysis and feedbacks, and simplifies collaboration with medical experts.

Other criterion are directly linked to the targeted monitoring tasks. For example physiological parameters monitoring is almost uniquely feasible with wearable sensors and fall detection is hardly achievable with only binary sensors (see next Section). More generally the *signal type* is important to consider for knowing which kind of signal processing to implement and features to compute, as well as *environmental interferences* like occlusions or signal distortions. This last environmental criterion and the *spatial information* are directly ruling the type of monitoring tasks that are achievable and which kind of algorithm is the most suited. For instance monitoring tasks requiring people tracking can not be done with a very low *spatial information* as it is the case with a unique microphone. Considering *environmental interferences*, they imply scalability issues relatively to the context and require robust and adaptable algorithms or massive data in various contexts for model training.

1.2 Health monitoring detection tasks

Depending on the patients to monitor, their living environment and the kind of sensors that are used, different detection tasks are achievable and targeted.

Physiological signals monitoring

Some monitoring systems focus directly on some physiological parameters coming from the patients body micro-signals. These kinds of detection regroup, for instance, respiratory and heart rates [28], blood pressure or sleep phase recognition. As they relate to internal body mechanisms that often entail almost no body motion, they usually require very sensitive sensors. These sensors are usually relatively close to the body, like wearable ones, or fixed and restricted location sensors that are meant to be almost in direct contact with the body, like bed mattress. For instance [157] proposes a complete framework with non-wearable sensors to perform ballistocardiography (heart beat forces and respiration), electrocardiography (heart beat rate) and photoplethysmography (blood oxygen and pressure). For that they use respectively a very sensible air mattress, capacitive electrodes on chairs and toilets and a light pulse emitter on chairs.

Physiological parameters monitoring is now studied for more than a decade by Emfit company, which is the manufacturer of the piezoelectric sensor used by Tarkett, and several works use this sensor in wearable and non-wearable devices for monitoring heart activity and respiration. For example [207] uses it in belts for measuring respiratory rate, the same sensor is embedded in a wheelchair [199] or usual chairs [131, 132] for ballistocardiography, and it can also be used for detecting epilepsy during the sleep [197].

Gait and posture analysis

Physiological parameters are not the only indicators that can give insights about patient health state, studying dynamics of different body parts while moving is also relevant for this task. Walking is one of the most repeated physical activity in everyday life and it involves complex local motions and coordination of every part of the body. Since several decades numerous works aim at characterizing elderly gait, extracting patterns and physical indicators (walk speed and regularity, left/right legs balance, ground reaction forces, etc...), and detecting disturbances and abnormalities [150, 185, 226]. These gait markers are known to be significant for assessing fall risks [37, 116] , for characterizing particular elderly pathologies like Parkinson disease [76] and even for detecting mood disorders [73].

Several recent surveys on gait analysis related researches describe existing used technologies [245], clinical applications [181] and discriminant features [58]. As the whole body motion is involved in gait analysis, the majority of existing methods use either wearable sensors [70] or cameras [224] but it is also feasible with other kinds of sensor. Indeed as previously evoked ambient sensors, like floor sensors [171, 241] or radio frequency based sensors [121, 232] are also used for gait analysis, as long as they can provide a precise enough spatial information.

Similarly, studying human posture in general during daily life activities can also bring indicators about patient health state. Thus some recent works focus on posture

recognition for elderly monitoring perspectives. For example some existing works use image processing knowledge to extract 3d posture models from videos for elderly monitoring [78, 284], but it is also possible to deal with this application using wearable device [161, 246] or even a panel of RFID tags [271]. One direct advantage of these approaches is that they also allow fall detection through the recognition of the "lying on the ground" posture [78, 273].

Fall detection

Fall detection is one of the main goals of monitoring systems for elderly. Indeed for older people falls are more likely to happen and to result in hospitalization or even death, especially when falls are not detected rapidly. Moreover, fall causes and risk factors are now well documented by medical community, as well as methods to prevent falls depending on patients diagnosis [210]. That aspect makes automatic fall detection and prevention a priority in the healthcare field and a very active research topic. Several surveys present issues, trends and challenges in fall detection [84, 125, 261], and compare different methods and approaches [192, 195, 223, 237].

There are two main ways to detect a fall : based on how the body is moving during the fall itself or based on the body posture on the floor just after the fall. That is why any sensor that can give body motion information or that has accurate enough spatial information to extract the body posture is useful for fall detection. Then, vision-based [281] and inertial wearable sensors [72] are commonly used for this task. However, even if it is less human interpretable and it may require more processing, several methods using other kinds of sensor are used for fall detection[154, 283].

Designing fall detection systems with real-time alarms is vital as "western" populations are getting older, falls are one of the major elderly unnatural death cause and even specialized nursing homes can not provide staff for continuous surveillance for every patient all the time. Progressing beyond triggering fall alarms would be to avoid these dangerous elderly falls, for this reason fall detection researches are progressively enlarging their scope into fall prevention [47, 208]. Nevertheless, because of the relative rarity of the fall event and for privacy reasons, fall datasets are still not massively publicly available [138].

Routine activity recognition, wellness assessment and anomaly detection

To prevent any elderly pathology or the fall risk, real-time detection alone is often not enough. Long term evolution of physical indicators, like gait or posture, or more generally changes in patient habits, represent indeed precious information for health risks prevention.

Depending on the segmentation of patient activities and their living environment, several sensors are used to extract patient routines from activity recognition. Whether it is based on vision sensors [186], binary sensors [213, 243] or hybrid sensor systems [82, 100], there are some similarities in the various approaches that aim at analyzing human habits. The first common step consists in the recognition of several daily life activities, either directly done by some binary sensors installed in various equipment [48] (sink, fridge, bed, chairs, etc...) or by applying a classification algorithm from other sensors [158]. Then the time spent on each activity is estimated to generate activity sequences with their associated durations. These sequences are used to train short

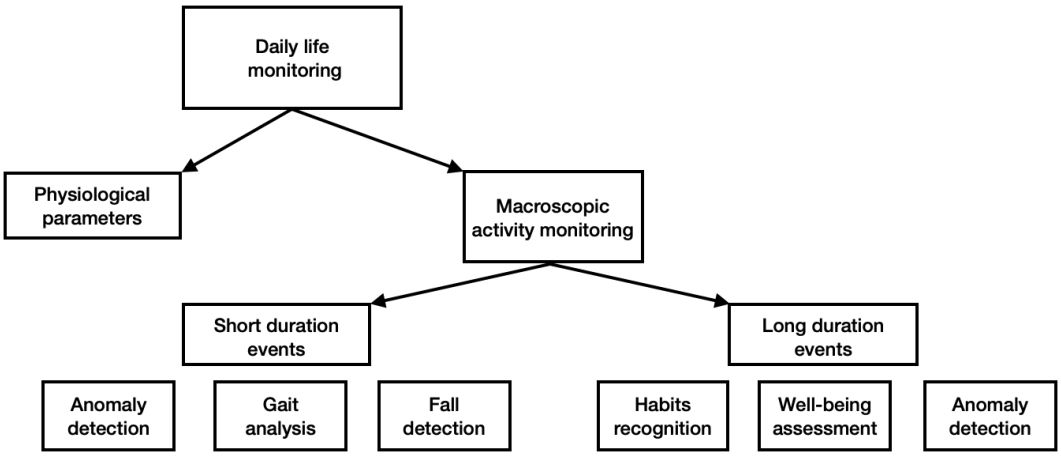


Figure 1.1: Taxonomy of daily life activities monitoring by detection tasks

or long term memory machine learning models that are sometimes represented as oriented graphs like hidden markov models and their variants (semi markovian models, hierarchical markovian models) [243] or more generally dynamic bayesian networks [81].

Although these approaches may need several sensors and a lot of processing and data, they are powerful for health monitoring in a daily life environment. Indeed, once a reliable model for habits extraction is obtained, it is then feasible to detect any abnormality by defining and measuring a deviation from these habit models [39, 41, 187, 213]. Moreover extracted activity sequences are also interpretable for medical experts which is an important advantage for health monitoring systems [42].

Detection tasks in Tarkett Floor in Motion Care application

Based on these main detection tasks for health monitoring, all the different monitoring applications can be categorized as presented in Figure 1.1. Health monitoring systems are designed either for controlling the internal functions of the body or more to observe the external macroscopic activity of the person. The first group is physiological parameters monitoring and is mainly used temporarily for people with specific pathologies, with wearable sensors or sensors installed inside objects that are in contact with the body (for example beds or chairs). People macroscopic activity monitoring is more adapted for daily life monitoring which can then be divided in two kinds of detection tasks : short duration or long duration events. Gait and falls are by far the two short duration events presenting the most interest for elderly monitoring. Long duration events regroup a wide variety of daily life events (sleeping, bathing, cooking, etc...) and the goal of detecting those is often to extract inhabitant habits, sometime attempting to assess person's well being. Considering anomaly detection these two event detection groups are concerned because anomalies can be either short in time, like a balance loss, or long like habit changes.

Tarkett elderly monitoring application, namely *Floor In Motion Care* (FIM Care), is currently proposing three main monitoring tasks : fall detection, detection of the activation or not of living space areas (see Section 4.2) and entry/exit detection. Nevertheless, as Tarkett has the tools for it, other topics that are presented later, like gait analysis and

habits characterization are also studied in parallel for future improvements. Unlike Emfit applications [131, 132, 207], Tarkett FIM Care system intends to monitor large living areas and are not targeting physiological parameters but more large scale body motion indicators. Next sections of this chapter are dedicated to present this system, to describe physical aspects behind Tarkett's sensing system, from pressure forces to processed output signals, and explain why seeked detection tasks are different from the ones proposed by Emfit company.

2 Tarkett piezoelectric smartfloor

2.1 Emfit sensor

Tarkett's smart floor system relies on a very sensible piezoelectric sensor, initially designed by the company Emfit to gather physiological features, like heart beats and breath, for example while laying on a mattress [197] or a chair [132] containing it. Among all different families (crystals, ceramics, etc..) of piezoelectric materials, a thin and flexible polypropylen electret is used inside Emfit sensor. It is a polymer foam where microscopic cavities are permanently polarized by applying a powerful electric field [189].

To obtain the final sensor, this polymer foam goes directly between two electrical conducing aluminium layers (one for the electrical ground and one for the signal output). These are surrounded by two insulating PET layers to isolate the electrical part of the sensor. Then, another aluminium and PET layer is added on the top to shield the sensor from possible external electric and magnetic parasites. The final product provided by the Emfit company is a sensor taking the form of 0.3 mm thin bands with a width of 60 cm. The simplest way to cover any area with this sensor is to cut several pieces of this 60 cm width band and to connect them in series to get one signal over the whole area.

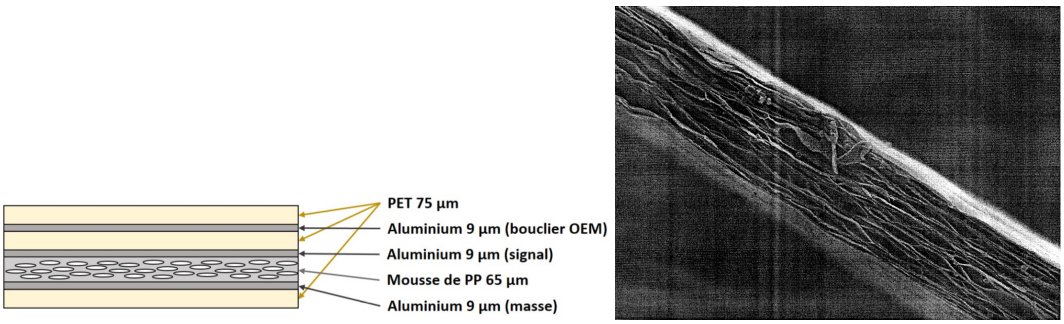


Figure 1.2: Structure and layers of Emfit piezo-electric sensor and microscopic view of the internal PP moss [189].

Each micro cavity in the material is permanently charged during the manufacturing process. While the piezoelectric film is compressed, these cavities thickness changes inducing a local charge displacement that spreads over the material to the output electrode. So that is how the piezoelectric effect is generated for this sensor, by creating dynamic currents that are approximately proportional to the total sensor's thickness

variation. This means, as it is specified in [228], that this kind of sensor is not suited for static but for dynamic measures and it is explained in details in the following Section.

2.2 A signal based pressure forces variation

As the purpose of this technology is to monitor elderly health state by analyzing mainly their physical activities, it is necessary to understand how the sensor output signal is linked to mechanical events that are occurring on top of it. Wearable motion sensors as accelerometers and gyroscopes are helpful for detecting some disabilities in the gait for example, and these kinds of motion information are correlated to pressure forces that are applied on the floor. In order to explore the links between floor sensor's outputs and other mechanical information, a simplified model of this sensor's physics is presented in the following.

Piezoelectricity is defined by two complementary physical phenomenon : it consists of a material that produces an electric signal while confronted to a mechanical stress (direct effect) and that inversely changes its shape while confronted to an electric field (inverse effect). Several physical variables are intervening in any piezoelectric phenomenon : the mechanical stress T (in $N.m^{-2}$) applied on the material, the electric field E (in $V.m^{-1}$), the relative deformation S (no unit) and the electric induction D (in $C.m^{-2}$). They are represented by tensors that interact in 4 main relations in the linear approximation theory with tensor coefficients characterizing the electro-mechanical properties of the material, the reader can find more details in [267]. As explained in [217], the equation traducing the main contribution in the direct piezoelectric effect is the relation between the mechanical stress T and the electric induction D is :

$$D = dT + \epsilon^T E \simeq dT \quad , \quad \begin{pmatrix} D_1 \\ D_2 \\ D_3 \end{pmatrix} \simeq \begin{pmatrix} 0 & 0 & 0 & 0 & d_{15} & 0 \\ 0 & 0 & 0 & d_{24} & 0 & 0 \\ d_{31} & d_{32} & d_{33} & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \end{pmatrix} \tag{1.1}$$

with ϵ and d two tensors respectively representing the electric permittivity and the piezoelectric factor between charges displacement and pressure forces. As the sensor is not supposed to be subjected to any external electric field, the first approximation consists of neglecting this part in equation 1.1. The second approximation concerns the matrix form equation linking a mechanical stress applied according to normal vectors of an infinitesimal cubic volume element (that is why T has a dimension of 6) to the displacement D of charges that can follow 3 axes. As the sensor is flat and inserted in the flooring, the main part of mechanical stresses are vertical up-to-down forces, simplifying the previous equation into :

$$D_3 \simeq d_{33} T_3 \tag{1.2}$$

This equation describes the approximate proportionality link between the surface pressure forces applied on the sensor and the amount of charge coming in and out of it. The d_{33} factor is then representing the sensitivity of the sensor. If the sensor is isolated (disconnected from any circuit) these charges accumulate in one of the two surrounding

aluminium plates (represented in Figure 1.2) creating a voltage difference between the two electrodes, which makes the sensor behave as a capacitor C_p. In addition electrodes can not stay charged forever and the voltage difference diminish progressively if no change occurs in the mechanical stress. This internal charge dissipation phenomenon can be modeled as a parallel resistor R_p, also called "leakage path". Then there are two ways of modeling a piezoelectric sensor as part of an electric circuit [228], depending on if it considered as a charge source or a voltage source:

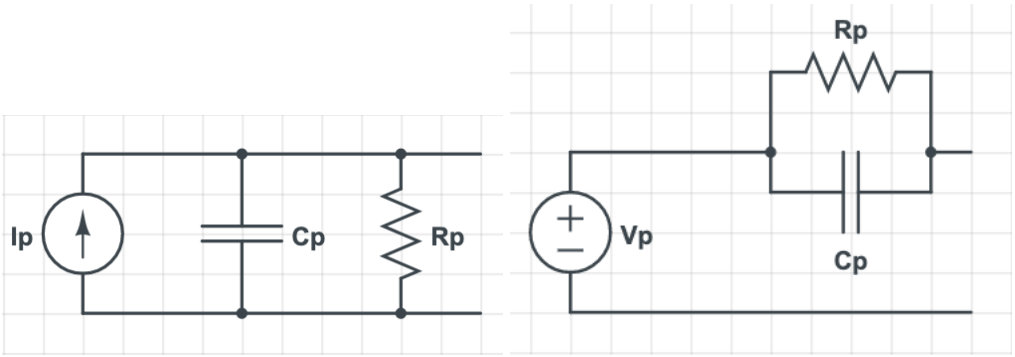


Figure 1.3: Charge and voltage source models of the piezo-electric sensor.

The closest model to sensor's signal generation is the charge source model, because local charge displacement are directly produced by pressure forces, following equation 1.2. This electronic model is ruled by the linear differential equation 1.3 with q_p the amount of produced charge and V_c the voltage between sensor's electrodes :

$$\frac{dq_p(t)}{dt} = i_p(t) = \frac{V_c(t)}{R_p} + C_p \frac{dV_c(t)}{dt}. \tag{1.3}$$

From equation 1.2, we can deduce that, with a variation of pressure forces ΔF, the total charges displaced are equal to $\Delta q = d_{33}\Delta F$. Thus no mechanical stress variation results in no current generation and then a voltage following the homogeneous differential equation solution : $V_c^{(H)}(t) = V_c^{(H)}(0)e^{-\omega_p t}$ with $\omega_p = \frac{1}{\tau_p} = \frac{1}{R_p C_p}$, and there are several important implications on the generated signal.

Firstly, it means that this sensor is not suited for static measure application but for dynamic ones. Indeed, an object or a person can be detected only while moving on the sensor, regardless of its weight, volume or surface contact with the floor. This also implies that weight assessment, which can be easily done with other kinds of floor sensor, is not directly feasible with this one. Secondly, while no dynamic activity is happening on the sensor a *relaxation* pattern depending on the time constant $\tau_p = R_p C_p$ is observable, which is comparable to a capacitor discharge behavior. As explained in Section 3.2, other time constant are also induced by the electronic signal conditioning montage but even if the relaxation time is variable it happens necessarily. Oscilloscope acquisitions obtained with a compression device show clearly this behavior in Figure 1.4. This relaxation phenomenon can be useful for some detection tasks as it is described in Appendix B.

3 Electronic signal acquisition and pre-processing

3.1 From micro-charge signal to time series

Signal amplification

In order to be able to process sensor's signal, the charge signal output has to be converted into a voltage signal. Moreover the signal coming directly from the sensor is hardly measurable because of its very small amplitude. Indeed, [217] approximates the sensor's sensitivity to : $d_{33} \simeq 234pC.N^{-1}$ and we can consider that normal event applied forces are below 2000 Newton (for instance [71] observed ground reaction forces during fall arrest going up to $1200N$). This means that the charge signal coming from the floor sensor has a range of $0.5\mu C$, which requires to be amplified before any usual signal processing operation.

There are two ways of amplifying this kind of signal, the voltage mode amplifier and the charge mode amplifier [228]. In the literature, it is advised to consider the voltage amplifier if it is directly connected to the sensor and the charge amplifier if there are wires in between, and thus because of wires capacitance. This is the choice made by Tarkett for signal amplification for its different versions of the signal conditioning circuit. Another important aspect of amplifying the signal is that it needs an electric supply. It implies that the output of the amplifier is not centered to zero and has a constant voltage component known as the direct current (DC) value.The stability of this constant voltage is crucial firstly to avoid signal's centering problem but more importantly because drifts of this value can result in overflows that destroy completely signal's shape. The main obstacle to this stability is the leakage resistance of the sensor that can be worsened by sensor's installation operations (see Section 3.4) and Tarkett observed that only symmetric supply amplifier are robust to this issue, as illustrated in Figure 1.4. Indeed there are two types of charge amplifier's supply : the single supply or the symmetric supply (in our case one entry of a $3.3V$ voltage or two entries of respectively $\pm 3.3V$). This is one of the differences between the first and the second version of Tarkett's electronic box (see Section 3.3)

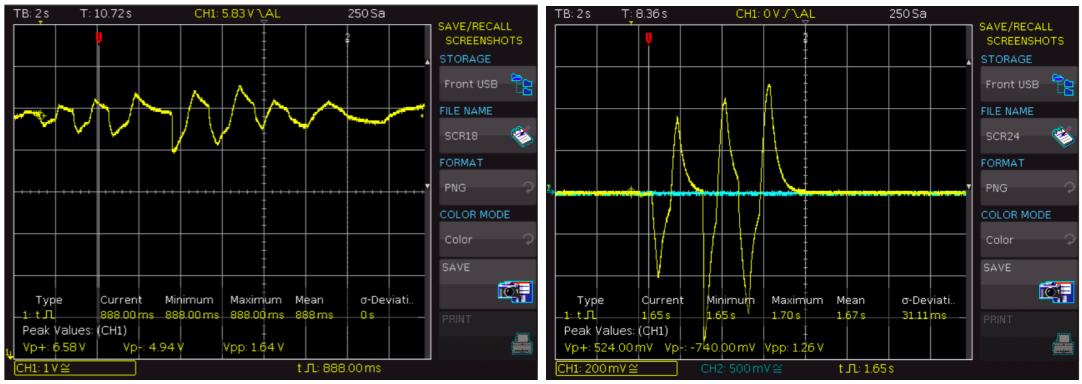


Figure 1.4: Difference between single (left) and symmetric (right) supply amplifier with an artificial leakage resistance of $1M\Omega$.

Analog/Digital conversion and digital signal processing

Once the sensor signal is amplified, it is an analog voltage signal that still requires to be converted into a digital signal by an analog-digital converter (ADC) for being processed by any computing unit. The choice of ADC parameters is a crucial part in sensor's signal processing because of two main aspects :

- Firstly, the ADC sampling frequency f_s restricts the detectable event frequency range to $[0, f_s/2]$, because of the Shannon's law.
- Secondly, the ADC numeric resolution determines directly the precision of the obtained digital signal.

The higher these two ADC features are and obviously the better is the sensor's signal quality. But they are costly and they also require associated CPU resources for this signal quality to be fully exploited. Having in mind this compromise, Tarkett decided to fix the ADC sampling frequency at $f_s = 100Hz$ and the numeric resolution at 12 bits. It means that there are exactly 4096 possible values for the digital signal and observable frequencies are limited to $50Hz$.

3.2 Signal conditioning and spectral domain

A charge amplifier is used by Tarkett to convert small amounts of charge coming from the sensor into a usable voltage. This amplifier works with an electric montage known as "pseudo-integrator", composed by one capacitor and two resistors. C_2 capacitor is used to accumulate charge coming from the sensor, allowing the effect of voltage "integrator". R_1 resistor protects the amplifier from electrostatic discharge while R_2 protects it from saturation. This montage is an active low-pass filter.

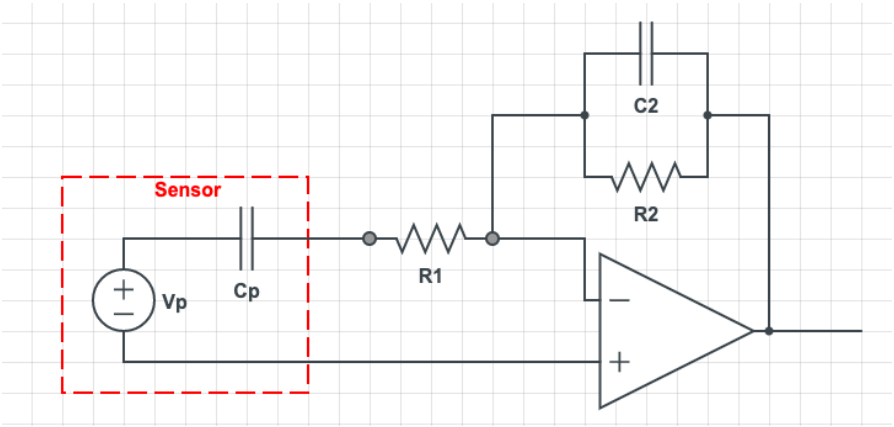


Figure 1.5: Piezoelectric sensor as a voltage source model with the charge amplifier electronic montage.

In order to determine how sensor's signal is affected by this electronic montage, the voltage source model of Figure 1.3 is used, but as the R_p leakage resistance is variable and depends additionally on manufacturing and installation imperfections (see Section 3.4), it is supposed there infinite which is an ideal situation. Usual method for describing the impact of the electronics on the signal is to compare a theoretic sensor's

voltage input $V_i(\omega)$, for a periodic signal of pulsation ω, with the corresponding theoretic output voltage $V_o(\omega)$ after this electronic montage. The ratio $H(\omega) = \frac{V_o(\omega)}{V_i(\omega)}$ is known as the *transfer function* of the system and it gives informations about how the signal is affected, in terms of amplitude and phase, from the sensor to the output of the amplifier circuit.

$$H(\omega) = -\frac{j\omega R_2 C_p}{1 + j\omega(R_2C_2 + R_1C_p) + \omega^2(R_2C_2R_1C_p)} \tag{1.4}$$

Equation 1.4 corresponds to the transfer function of a second order band-pass filter that can be written in the canonical form as :

$$\begin{aligned} H(\omega) &= \frac{H_0}{1 + jQ(\frac{\omega}{\omega_0} - \frac{\omega_0}{\omega})}, \\ \text{with } H_0 &= -\frac{R_2C_p}{R_1C_p + R_2C_2}, Q = \frac{\sqrt{R_1C_pR_2C_2}}{R_1C_p + R_2C_2}, \omega_0 = \frac{1}{\sqrt{R_1C_pR_2C_2}} \end{aligned} . \tag{1.5}$$

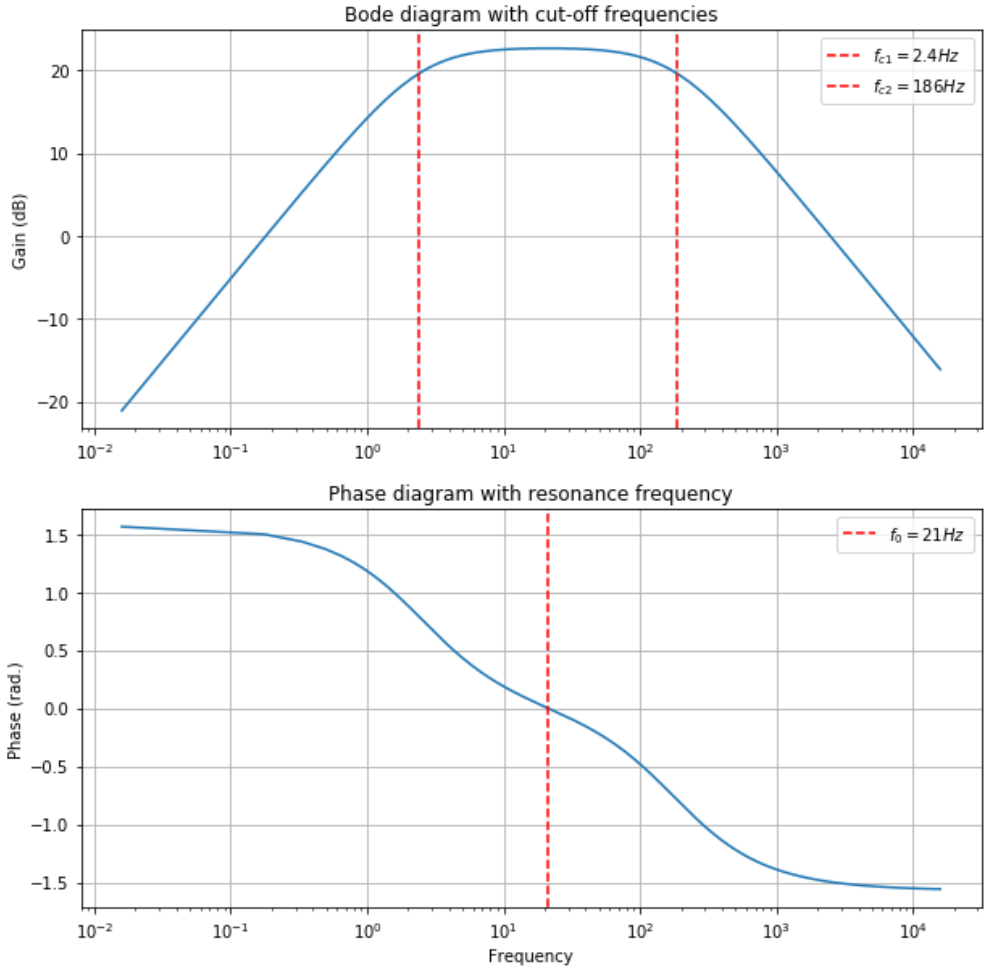


Figure 1.6: Example of phase and Bode diagram of signal conditioning circuit's transfer function for a $20m^2$ sensor.

H_0 is the maximum gain value reached for resonance pulsation ω_0. Q is the quality factor representing the narrowness of the frequency band-width $\Delta\omega = \omega_{c2} - \omega_{c1}$ of the filter by $\frac{\Delta\omega}{\omega_0} = \frac{1}{Q}$, with ω_{c1} and ω_{c2} the cut-off pulsations defined by the relation : $|H(\omega_c)| = \frac{|H_0|}{\sqrt{2}}$ that has two solutions of the form : $\omega_c = \frac{\omega_0}{2Q}(\sqrt{4Q^2+1} \pm 1)$.

As the leakage path resistance is neglected in this model the only variable value is the capacitance of the sensor C_p. Indeed it depends on sensor's area and its surface capacitance density is estimated around $22pF.cm^{-2}$ [217]. For an usual room area of $20m^2$ this corresponds to a capacitance $C_p \simeq 4.4\mu F$. The rest of electronic components are fixed at following values : $R_1 = 200\Omega$, $R_2 = 200k\Omega$ and $C_2 = 320nF$. These parameters give a maximum gain of $|H_0| = 13.6$ and cut-off frequencies $[f_{c1}, f_{c2}] = [2.4Hz, 186Hz]$, phase and Bode diagrams are presented in Figure 1.6.

The first remark is that main physiological parameters like heart and respiratory rates are signals with spectra below this lower cut-off frequency of $2.4Hz$. This means that, even without considering any noise issue, with this area size and this electronic montage, measuring physiological parameters of someone lying on the sensor is hardly achievable. Secondly the upper cut-off frequency is highly sensible to sensor's capacitance and then to sensor's area. To give an insight about this, the maximum observable frequency of $50Hz$ allowed by the ADC is reached by the upper cut-off frequency around a sensor's area of $80m^2$ (implying $Cp \simeq 17.6\mu F$), meaning that it is impossible to go over this size without degrading importantly the signal's frequency range.

3.3 Various electronic devices in experimental and real environments

As previously explained, the output of the Emfit piezoelectric sensor is hardly exploitable directly due to the very small amplitude of signals. Knowing the minimal requirements in terms of signal conditioning for this sensor, each component of the signal acquisition electronics has to be rigorously chosen, as well for their values as their quality, depending on the intended use. For example in [132], authors decide to add another low-pass filter circuit as they are only interested in low frequencies events (heart and respiratory rates).

This means that different signal acquisition electronics leads to different shapes of data. Main priorities while designing signal conditioning electronics can be drastically different in experimental or real environments. For experimental purposes one would want to get a signal shape as close as possible to the sensor's output, in order to be able to explore all the range of application that are theoretically achievable.

For that, an electronic device that captures the widest possible range of frequencies is needed, as well as a scalable one regarding signal amplitude to be able to experiment several sizes of sensor. Moreover, to be as close as possible to the analog sensor's output and to capture highest frequencies a high resolution ADC is also needed. Finally, as experimental devices are designed for very restricted use and are not meant to be chain produced, high quality electronic components inducing the least possible noise are preferable. Here are presented several devices used by Tarkett from the first experimental steps to their final product implemented in real environments.

Experimental environment electronic devices

Most controlled low level sensor-based and electronic-based experiments on the system are done using a oscilloscope. It needs external signal conditioning as sensor's output signal is too low but it allows to observe directly impacts of changing electronics components on several electronic measures. It is mainly used on small sections of sensor for sensitivity and basic response measurement with a precise pressure application device of brand Inströn. Oscilloscope is usually employed to observe sensors that are not even installed in the flooring and no database has been recorded with this device.

The second device used for experimental purposes is the DS1103 prototyping panel of dSpace brand for analog signals. It also needs external signal conditioning as it is not sensible enough to deal with very low amplitude currents such as those produced by the sensor. It allows real time signal visualization through a software named ControlDesk and signal recording and processing are done using Matlab and Simulink, which is a graphical programming software. Advantages of this device are that it is simple to test several external signal conditioning components to observe directly their influence on signals, an internal ADC is programmable which allows observations with a wider range of frequencies and the ability to test in real time the effect of digital processing and various models with Matlab/Simulink. Two databases have been recorded using this device in a nursing home installation described in Figure 1.8 with an electronic card for signal conditioning that is comparable to the ebox v1 circuit.

The only measurement device that is sensible enough to observe sensor's output without any signal conditioning circuit is the Electrometer (model B2987A of Keysight Technologies manufacturer). It is a costly professional device capable of measuring directly very low charge signals between $1fC$ and $2\mu C$ which is precisely the kind of output coming from the sensor. The main advantage of this device is that it can perform measurements that are very close to sensor's response, independently to any amplifying circuit. It also allows real time visualization and recording as well high frequency measures (up to $20kHz$). One database has been recorded using this device in experimental conditions with simulated events, to study the feasibility of detecting low amplitude moves of people lying on the floor.

Real environment electronic devices

Electronic devices used in real environment are largely produced, installed in every room equipped by Tarkett monitoring system and their function is to process sensor's signal, to apply several real time detection algorithms and to store signals in order to send them to servers if needed. Tarkett designed two different versions of these processing units, namely *ebox v*1 and *ebox v*2.

- **ebox v**1 **:** This device possess 8 input channels to connect wires coming from different sensor areas and every channel is linked to the following : an amplifying circuit with a single 3.3V supply as described in 1.5 and an ADC with 12 bits resolution and 100 Hz sampling frequency. Then the 8 digital inputs obtained after these electronic stages are entering a computing unit with 256 kB ROM, 16 kB RAM and 40 MIPS.

- **ebox v**2 **:** The second electronic box version has been designed for several purposes : to avoid some malfunctions, enhance signal quality and increase

computing resources. For that, the single supply amplifier is replaced by a symmetric supply amplifier (see 1.4), another amplifying circuit stage is added to the first one to treat low amplitude signals on each of the 8 channels, what makes 16 signal inputs to the computing unit (8 amplified once and 8 amplified twice). Finally the new design computing unit capabilities are : 256 MB for ROM, 256 MB for RAM and 600 MIPS.

These electronic boxes need to be connected to electric supply and internet network. One hour signal records are stored automatically in their memory which is refreshed every hour, and Tarkett can distantly launch the command to download these records. Moreover, several useful informations are provided to medical staff through Tarkett's Web application (see Figure 1.9), like activity periods or entries/exits, are also stored every hour and sent through internet.

Usage	Device	Signal Cond.	Feq.	ROM	RAM	CPU
Unique	Oscilloscope	No	>10kHz	-	-	-
	dSpace	No	>10kHz	-	-	-
	Electrometer	Yes	>10kHz	-	-	-
Industrially Produced	Ebox v1	Yes	100Hz	256 kB	16 kB	40 MIPS
	Ebox v2	Yes	100Hz	256 MB	256 MB	600 MIPS

Table 1.2: Electronic signal processing devices used by Tarkett.

Table 1.2 summarizes all electronic devices used by Tarkett for experimental and real measurements. Oscilloscope, dSpace and Electrometer are unique devices whereas ebox v1 and v2 are industrially produced. Most important differences between ebox v1 and v2 are computing resources. ROM is the static memory space where models and routine programs are saved, RAM is the dynamic memory space where temporary computations are saved and the MIPS value means the amount of operations (in millions) the CPU is able to execute in one second. These values represent practical bounds relative to model size or prediction time and machine learning approaches to take them into account is the topic of Chapter 4.

3.4 Sensitivity, variability and robustness of the sensor

Manufacturing variability

As mentioned in equation 1.2 the sensor's sensitivity is measured by the d_{33} piezoelectric parameter. The film sensor used by Tarkett in their smart flooring is provided by Emfit company in the form of hundreds meters long rolls. From one sensor roll to another, the mean sensitivity varies less than 5% but Tarkett observed that high variability are possible at rolls extremities that can reach 40%. One possible explanation is the transportation conditions (especially high pressure levels during extended period of time) and this variability forces Tarkett to get rid of several meters in each roll for its smart floor application. Nevertheless, in order to not waste too much sensor as it is costly, Tarkett's goal is to design monitoring algorithms that are still able to handle a sensitivity variability of 20%.

Installation variability

The inner sensitivity of the sensor is not the only aspect impacting signal's amplitude. As it is proportional to the mechanical stress exerted, it also depends on the rigidity of the flooring material put on top of the sensor film, which can be different for each nursing home. Moreover the signal amplitude is directly linked to physical parameters relative to the sensed body like its weight or the type of worn shoes. So each room installation possesses its own signal's amplitude for a given event and it is then crucial to be able to deal with a certain amount of amplitude variability.

Apart from sensor response amplitude, installation variability is observable in other important aspects. Indeed Tarkett smart floor installation process is not yet fully industrialized and two critical steps remain manually done : cutting sensor bands and connecting them in series. Maintenance feedbacks showed that if these two sensible operations were not executed properly some malfunctions, like signal drifts or overflows, can occur due to short-circuits or high external leakage resistance and not necessarily observable directly during installation tests (mainly because of micro-particle residuals and humidity accumulation). Finally, depending on installations, floor area size covered by sensors can vary, implying different sensor's area capacitance and then different signal behaviors as explained in Section 2.2

Signal noise

A last important aspect of signal's quality that can also vary between equipped rooms is the signal's noise. Tarkett observed a ratio of 5 between noisiest rooms and least noisy ones without any satisfactory explanation of these differences for now, but two experimental tests show that this noise is essentially coming from electronics. Firstly the Electrometer which is the most precise measurement instrument and the only one that does not need any external electronic processing shows negligible noise compared to electronic boxes (v1 or v2). Secondly noise acquisitions with electronic boxes show no noticeable difference while they are disconnect to any kind of input, proving that this noise is neither coming from the sensor itself or from any connector or wire. Indeed there is no ideal electronic component and each one of them might induce some electrical noise, especially the amplifier that needs a constant external electric supply which is possibly noisy itself. For this reason even if it can be costly, each electronic component of the signal conditioning circuit has to be chosen with great care in this kind of application.

4 Tarkett's monitoring system installations and datasets

In a few years Tarkett managed to install its floor based monitoring system in several elderly care institutions, several countries and with hundreds of equipped rooms. From the start of this project, Tarkett did not cease to innovate and improve progressively its system based on client feedbacks and experimental studies. These changes occurred in every level of the technology from the way to characterize, select and install the floor sensor, to the method of processing the signal electronically, upgrading from ebox v1 to ebox v2, and to gather data through Tarkett system network architecture. Finally algorithmic enhancements in the different levels of A.I. intervening in this monitoring

application are also concerned. This implies working with various sources of data, coming from different electronic processing devices (presented in the previous Section) but also from different environments and contexts. This Section presents the main databases used throughout this work with their characteristics and the conditions in which they were obtained.

4.1 Experimental databases

In this work we will refer to different existing datasets as "real" or "experimental" depending on whether they are composed by signals from real situations in elderly healthcare environments or not.

Simulated falls : This database is composed by signals of simulated daily life events (walking, sitting, moving objects, etc..) and simulated falls with different initial and final positions. They were generated by 28 employee volunteers aged from 25 to 45 years old that followed a precise protocol for each acquisition and the mean signal length is around 20 seconds. It has been done on a small reproduction of a basic Tarkett room (See Figure 1.7) at Tarkett R&D center and using ebox v1. The 742 acquisitions of this database are organised in two main labels are *Fall* (409 signals) and *Non-fall* (333 signals). That was the very first Tarkett's database created in 2015 and the initial fall detection model was trained with these data (see Section 2).

On floor activity : This database of 1332 events was recorded in 2017 by 28 volunteers (same as the previous one). The motivation was to enhance the fall detection task by training a classifier able to detect human activity while lying down on the floor. Indeed falls with relatively soft impacts are the hardest ones to detect and elderly are more likely to be lying on the ground in conscious state in these situations. 15 behaviors have been simulated, as presented in Table 2.1, trying to be representative of activities that are normal and other that are or not, lying on the ground. These 15 labels are grouped into 3 main labels : *Walking*, *On-Floor* and *Other* (low amplitude moves and silent events). These acquisitions were done with the Electrometer device in order to get very precise measurements and to have scalable data to different signal conditioning electronics, as explained in Section 3.2. The classification strategy on this database is described in Section 2.

4.2 FIM Care room installation

Real datasets are composed by nursing home data obtained either from elderly patient rooms or particular highly frequented common space sensor installation especially dedicated for recording walk signals in a corridor and sit/stand signals in front of a TV. This common space installation regroup indifferently events from elderly, medical staff or visitors, are also equipped with cameras for labeling and are recorded through dSpace device, whereas elderly patient room acquisitions are done with ebox v1 or ebox v2 and without any camera or other sensor for labeling.

As illustrated in Figure 1.7 a typical Tarkett system room installation is composed by three areas : the entry, the bathroom and the bedroom and an electronic box (v1 or v2). As the two ebox versions have 8 input channels an ebox can deal with maximum 8

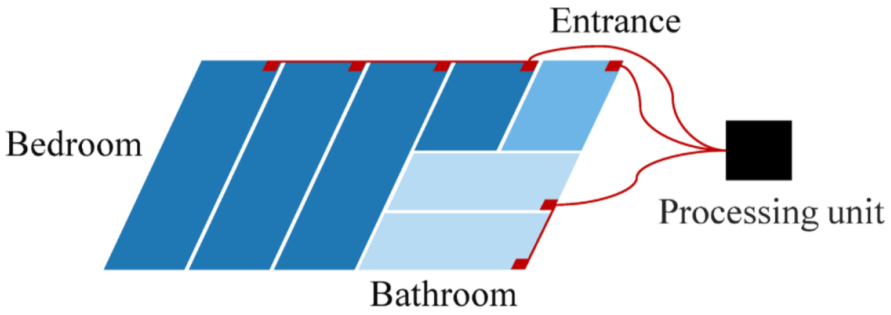


Figure 1.7: Basic Tarkett room installation for elderly monitoring and fall detection.

areas, so Tarkett also equipped some wider elderly apartment with more areas but the huge majority of nursing home rooms are only composed by these three areas. The entry is a section of a 60 cm band whereas other areas are formed with several of these sections connected in series by small electric connectors and linked to the ebox through wires inserted in wall plinths. Sensors are fully covered by the flooring and electronic boxes are usually placed in a cupboard in the room.

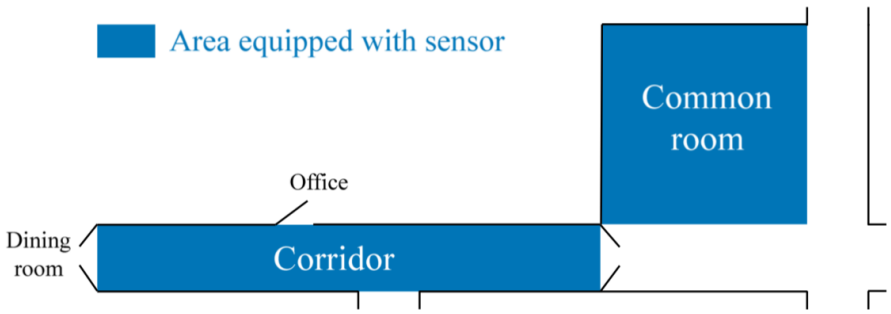


Figure 1.8: Tarkett corridor installation for walk analysis.

Another real condition installation in a nursing home is illustrated in Figure 1.8 mainly for gait analysis, it consists of a highly frequented corridor where elderly and medical staff walk everyday and a common TV room equipped with chairs for elderly people. Three cameras are located in these areas for labeling but it needs dSpace device use and technical experts to record floor sensor data. Two databases using this installation are presented in the following.

4.3 Real environment databases

Following databases are either obtained from real situations in the set-up presented in Figure 1.8 with cameras or blindly from elderly patient rooms comparable to the Figure 1.7 installation.

Real falls: This database of 2717 events was recorded between 2017 and 2018 in real environments from several retirement homes equipped with Tarkett's monitoring system. The motivation was to create a fall database more realistic than the previous simulated fall database, in order to confront these two datasets and to adapt detection algorithms taking into account these differences. Every signal comes from extraction done on one hour signals of real elderly people in their room recorded with ebox

v1 and sent to Tarkett's servers. To be more representative of what happens in real conditions, this dataset is extremely imbalanced with 154 fall signals and others are randomly extracted non-fall events. Falls are either real falls detected by the fall detection algorithm implemented at the recording moment and confirmed by medical staff, or real falls that were not detected by the algorithm and then extracted manually after medical staff feedback. The procedure for non-fall extraction was to first select one hour recorded signals that were confirmed to not contain any fall, and then to extract sub-signals corresponding to activity moments.

Real walks : This database of 146 signals was recorded in 2017 in a retirement home equipped with Tarkett's monitoring system. A particular installation has been deployed for this purpose, consisting in several bands of sensor disposed in an important traffic corridor of the nursing home. Each band was linked with electronic wires to dSpace device and passing through a signal conditioning circuit comparable to the one used in ebox v1. Signals labeling has been done manually by experts using two cameras located at the corridor's extremities. This dataset gathers different kinds of walk of elderly patients and medical staff members organized into 8 labels : medical staff walk, elderly walk, multiple walks, manual wheelchair, pushed wheelchair, electric wheelchair, walk with cart and other.

Medical diagnosis motion data : This database of 904 signals was recorded in 2017 by medical experts from a motion analysis perspective. It is divided in two main parts : *real environment* signals with freely moving people and *experimental conditions* signals with people asked to follow precise protocols. Every signal of these two environment was recorded using the dSpace device and a signal conditioning circuit comparable to the one used in ebox v1. Thus 644 walk acquisitions were done in the same corridor and conditions as previously described real walks database, with a multidimensional labeling : firstly whether it is a medical staff walk, a patient walk or a visitor walk; secondly whether it is an equipment free walk, a walk with an equipment (in this case with details on the equipment) or a multiple people walk; thirdly, for patient walk, whether is helped by another person or not. Considering *experimental conditions* scenarios, 110 sit-to-stand acquisitions (from a bed or a chair) and 150 wheelchair acquisitions were recorded with 10 Tarkett employees. The particularity of the experimental part of the dataset is its rich contextual information about people executing the moves (age, gender, approximate weight and height) and all the different motion phases details.

Ebox v2 acquisitions : This database is only composed by raw signals of elderly patient daily activities in their room. It is mainly used by Tarkett to compare signals with ebox v1 in terms of amplitude, precision, quality and defaults. It is necessary for future developments of the monitoring system but, as no further treatment have been applied on raw signals, this database is not directly exploitable in our studies on predictive models.

Events	Device	Env.	N° labels	Size	Duration
Simulated falls	Ebox v1	Exp.	2	742	20s
Real falls	Ebox v1	Real	2	2717	20s
On-Floor	Electrometer	Exp.	15 (3 main)	1332	50s
Medical diagnosis	dSpace	Both	$(3,3,2)+2$	904	30s
Daily life	Ebox v2	Exp.	-	-	-
Real walks	dSpace	Real	8	146	10s

Table 1.3: Tarkett experimental and real databases for the elderly monitoring application.

5 Technological constraints for industrial monitoring systems

This chapter developed a theoretical model explaining how mechanical stress forces applied on Tarkett's sensor are converted into digital signal values after amplification by an electronic montage and an analog/digital converter, which enables simulations from pressure force data and makes easier to understand and work with data used in the rest of this thesis. It pointed out that several technological parameters can have a direct impact on the signal's nature and then on the ability to perform some detection tasks. In particular Tarkett uses a sensor designed for dynamic events monitoring and then is not suited for certain static application like direct weight measure [12, 43] or immobile object detection [38, 78]. This sensor has also a relaxation behavior after a mechanical stress, that is dependent on its area and its internal leakage resistance. Because of the electronic design detectable event frequencies are between $2.4Hz$ and $50Hz$ excluding some applications like physiological parameters measurement. In addition all the variety of measurement devices and databases has been described to present the overall context of this work on Tarkett's industrial monitoring system.

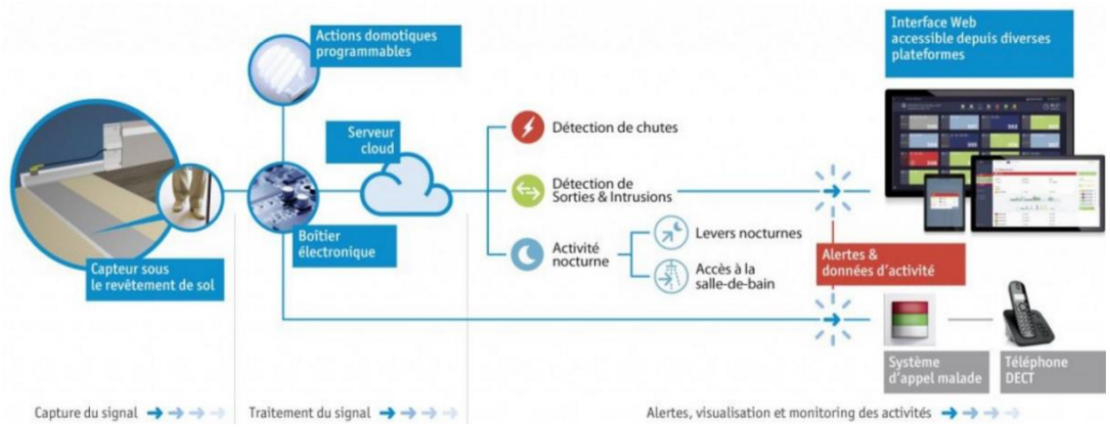


Figure 1.9: Tarkett monitoring Web application architecture.

5.1 Detection tasks

The goal of Tarkett is to provide a health monitoring system for elderly people able to handle several detection tasks from piezoelectric flooring sensor signals. The main

proposed detection tasks, given in Figure 1.9, are *fall detection*, *entry/exit detection* and *night activity detection*. Falls are obviously the event with the most impact on elderly health, with major injuries and even death risks, their detection is then the main priority. The ultimate purpose regarding this application would be to be able to assess fall risks in order to prevent them. Other tasks concern human activities without a lot of details, like room's entries and exits, or night activities. The main purpose being to distinguish among them abnormalities like intrusions and highly repeated wake-ups or bathroom accesses during the night.

Besides of what is proposed to the clients, other detection tasks are targeted. Some are directly necessary for the system maintenance as technological *malfunctions detection* (abnormally high noise, signal drifts, sensor's sensitivity issues). Others are intermediate tasks that can be useful for previous detection tasks or data labeling, as *patient/staff differentiation*, *gait recognition* or *people counting*. Finally Tarkett also seeks for long-term detection tasks to enhance its monitoring system services, as *patient habits characterization* and the ability to detect habit changes or anomalies, ultimately using that for health risks assessment.

5.2 Data sources

As described in Section 4, designing machine learning models for all these detection tasks entails dealing with data heterogeneity due to various data sources. There is firstly the variability coming from the sensor and the measurement devices themselves. There are some sensitivity variation between different samples of sensor and as explained in Section 2.2 different size of sensor (for different room sizes) implies different relaxation time constant. Signal processing also differ between experimental purposes devices and ebox v1 or ebox v2. So available signal data are subject to heterogeneity due to sensor and hardwares variability in terms of amplitude, but also frequency response and noise or signal distortions.

The other kind of data heterogeneity comes from the type of environment and context in which data are recorded. The most apparent context difference is between experimental data, with simulated events executed by people that are told to following a protocol, and real data. However even among real data there are some dissimilarities. Indeed different elderly healthcare institutions or different departments of one institution can regroup different kind of patient pathologies or diseases. Finally each patient has its own physical particularities, like age, gender, height and weight, moving equipment (cane, walker, wheelchair), that can impact the nature of signals.

5.3 Resource limitations

Every industrial application is subject to several kinds of resource limitations to work properly, whether it is financial resources, computing resources or even human resources. As previously explained the core of Tarkett's monitoring application relies on algorithms that are embedded in an electronic device installed in each room. This means that detection tasks are constrained by a limited prediction time especially for the ones supposed to trigger emergency alarms as fall detection. Moreover machine learning models used for these detection tasks and the associated needed computations are also subject to memory limitations. Depending on the version of electronic box

computational resources are different (see Table 1.2) and possible embedded algorithms differ. It is then crucial to take these limitations into account while designing machine learning models for Tarkett's monitoring application.

Data gathering capacities can not be unlimited either. Indeed it is possible to record remotely data measured by an electronic box through internet network into Tarkett's servers. But firstly, for a given elderly healthcare institution the amount of rooms sending signals at the same time is limited by the internet upload capacities of the institution. Furthermore data storage is costly and it is not affordable to record data from every equipped room, then Tarkett has to make the choice of which room's data to record and when. Finally, any recorded data arrives in the form of raw unlabeled signals and data labeling, either it is done by Tarkett employees or medical institution's staff, is costly in term of human resources which are also limited.

2

Time series representation and decision tree predictive models

1 Human activity time series representation

1.1 Feature extraction on time series : overview

Sensor's signal that Tarkett has to deal with is, as explained in previous Chapter, a continuous analog charge signal approximately proportional to pressure force variation applied on the floor. This signal is also electronically and digitally processed to obtain an amplified and filtered time series version of this source analog signal. Contrary to physically interpretable motion sensors as accelerometers or gyroscopes, these time series do not reflect directly the body motion on the sensor and the extraction of relevant features is necessary to perform complex detection tasks like fall detection or gait analysis.

At first sight, in general these computed features depend on the kind of sensor and its output which can be categorized according to data dimensionality. These sensor data are usually either images for camera based systems or one dimensional time series for other systems (see Table 1.1). All these different kind of data can be viewed as time series of varying dimensions, with camera videos that are matrices series or 3 dimensional tensors in the case of depth cameras [278], and non vision based sensors often deliver one dimensional time series on several channels. Considering binary sensor systems, extracted features are often close to direct sensors outputs. For example, in [239] the sequence of multiple binary sensors activation is directly used as features to train a hidden markov chain model while [213] uses short patterns of sensors activation to train a finite state automaton. These kind of systems can be viewed as boolean time series with numerous channels.

The number of channels and how much their corresponding time series are redundant is also important with regards to the richness and the reliability of the sensors signals information, thus the feature set has to be adapted to the multiplicity and redundancy of inputs. For example accelerometers often provide signals on 3 channels corresponding to 3 independent axes, but used in combination with gyroscopes or magnetometer these channels are correlated, which is used for better reliability [148, 222]. Employing correlated channels of time series can also be used to avoid obstruction issues for cameras for instance [81] or to deduce spatial information with microphones [154].

Furthermore desired detection tasks and used models determine also importantly the feature set, for example extracted features for intrusion detection might not be the same as for fall detection, and the feature extraction effort for a simple threshold based model might be higher than for a neural network, in which features would be computed intrinsically through first layers. So depending on these aspects, human activity monitoring applications often require their own particular set of features of adapted nature, size and complexity.

Usually, while sensor's output is not directly exploitable to extract relevant features for desired detection tasks, then computing a different data representation of time series is a common approach. Indeed to study frequency domain can be more relevant while attempting to detect short duration highly dynamic events with good signal resolution whereas autocorrelation computation or autoregressive models with wavelet decomposition would be more suited for events with repeated patterns [6, 119, 250].

For our monitoring application, we decided to investigate the widest features group

in experimental conditions in order to obtain an important variety with regards to the data representation, and then to study separately the selection of a sub-group of features, which is necessary to have a functional model in real conditions.

1.2 Time and frequency representations of the signal

Main monitoring tasks have to be executed in real time within an embedded system and therefore detection tasks have to be based on a small real time signal window of at most a few seconds. Then a signal representation is a function of this signal portion, preferably reversible, such as it conserves the main part of the original signal information, but providing emphasis on some time/frequency aspects.

The choice of a suited signal representation is an important step used to better point out events targeted by the desired detection tasks. This step determines the further feature extraction step and moreover increases the interpretability of the feature set. In this section is presented the time/frequency representations of the signal used for feature extraction.

1.2.1 Derivative and integral

Signal's derivative and integral are two representations that serve respectively to point out local or average behaviors of the signal, with the derivative that tends to sharpen variations and integral that tends to smooth the signal. As explained in the next section, all signal representations and features will be computed on a fixed length S sliding window, so in this case these representations are discrete and finite approximations defined for a signal vector $(s_0,...,s_{S-1})$ as follows :

$$\forall 0 \leq j < S, \quad D[j] = s_{j+1} - s_j, \quad I[j] = \sum_{k=0}^{j} s_k \quad .$$

Figure 2.1 illustrates different kind of event signals with their corresponding derivative and integral. In terms of physics the integral of the signal should be homogeneous to the total applied force on the sensor but the relaxation phenomenon compensates this integration and also permit the amplifier to not saturate. This phenomenon could help to emphasize spikes of numerous people on the sensor or while a pressure is brutally applied as in the case of a weight launch.

For these reasons, the derivative representation should intuitively be useful for detection of rapid high amplitude variations events like brutal falls whereas the integral quantifies the disproportion between positive and negative peaks. Thus relaxation phenomenon can be pointed out by the integral as it is observable for weight launches and moving or removing objects from the ground can lead to negative integral as it is observable on the chair move event.

1.2.2 Autocorrelation

Autocorrelation is the correlation of the signal with a delayed version of itself, defined as :

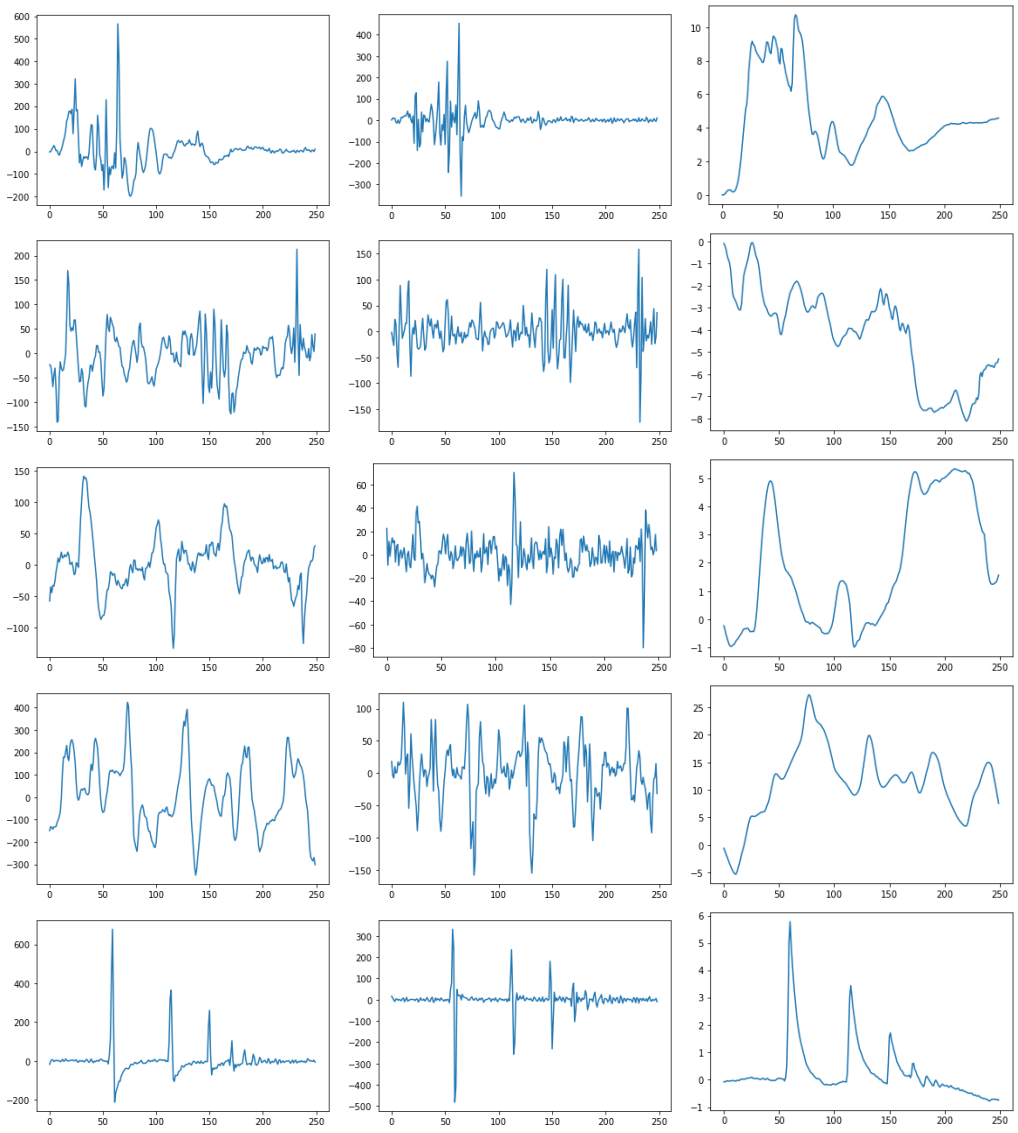


Figure 2.1: Various event signals with their derivatives and integrals.
First column represent original filtered signals, second one is the derivative and the third one is the integral. In row order these events correspond to a fall signal, a chair move signal, two different walk signals and repeated weight launches.

$$\forall 0 \leq j < S, \quad R[j] = \frac{1}{\sigma_s^2} \sum_{k=j}^{S} (s_k - \mu)(s_{k-j} - \mu) \quad ,$$

with μ the mean and σ_s^2 the variance of the signal.

This representation can be used to spot events with repeating patterns, such as walks, balancing on a chair or doing some physical exercise. Daily life events containing repeating patterns are common and so are several staff activities like room cleaning. Thus recognizing them can be useful to distinguish staff activities from patient activities. A work on patient/staff recognition based on walks is presented in Section 2.4. Indeed, the main event that can be efficiently characterized by autocorrelation is the walk and

this representation can theoretically serve to quantify some gait features like walk speed and regularity.

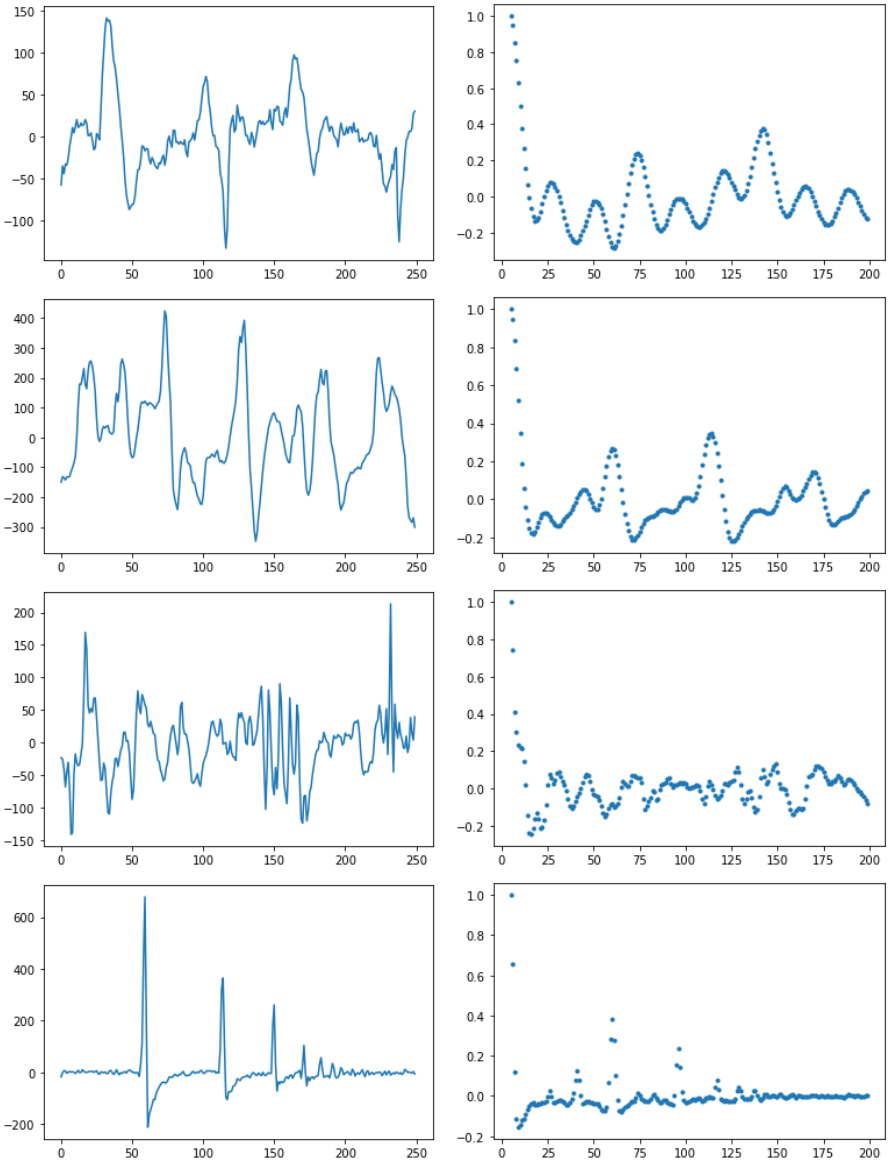


Figure 2.2: Example signals with their autocorrelation.
In row order these events correspond to two different walk signals, a chair move signal and repeated weight launches. As shown autocorrelation seems well suited for walk recognition.

1.2.3 Fourier transform and spectrogram

Fourier transform and spectrogram are used to study the spectral domain of event signals. They can be useful to characterize and discriminate highly dynamic events with rapid variations like human or object falls, or continuous stresses on the floor by objects like chair moves or wheelchairs. Indeed, as explained in Section 3.2, the lower

cut-off frequency of the electronic signal conditioning system is around $2.4Hz$ which is yet higher than a usual elderly walk.

These discrete Fourier transform and spectrogram are respectively defined as follows :

$$S[f] = \sum_{k=0}^{S-1} s_k e^{-\frac{2\pi i}{S} fk}, \quad W[j,f] = \sum_{k=j}^{S-1} s_k s_{k-j}^* e^{-\frac{2\pi i}{S} fk} \quad .$$

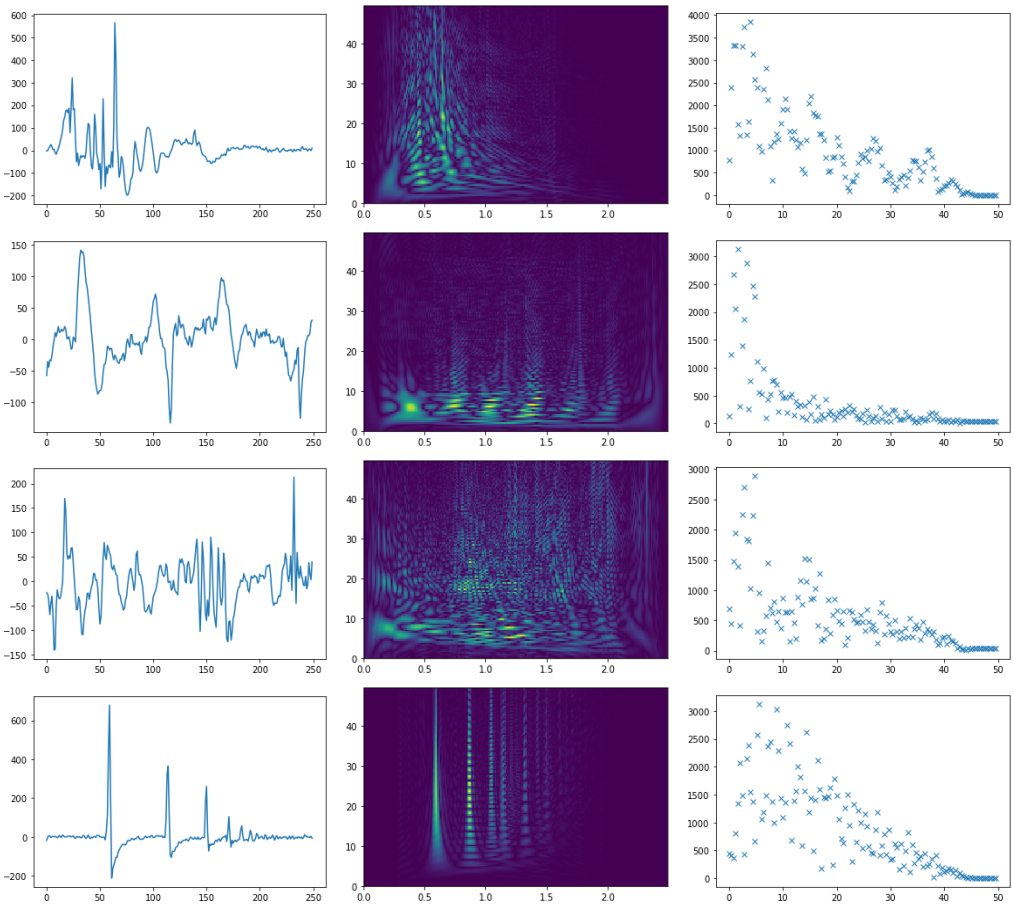


Figure 2.3: Frequency representations of various event signals.
The middle column represents Wigner-Ville spectrograms and last columns is the fast Fourier transform. In row order these events correspond to a fall, a walk signal, a chair move signal and repeated weight launches. For both these time series representations, the signal energy location concentration appear to be important to consider to perform event detection.

It can be pointed out that complex approximately repeated patterns can be clearly observed on spectrograms although being harder to emphasize using only Fourier transform. Moreover the energy concentration in areas of the spectrogram is an important event indicator, chaotic events on the contrary have tendency to create a dispersed energy fog in the spectrogram.

All these signal representations are computed essentially from the original signal (contrary to parametric representation like autoregressive models) so they contain the same information but expressed differently. Observations made on simple and intuitive

behaviors of each of these allowed to design adapted feature sets, grouped by signal representations and described in the next section.

1.3 Experimental feature set on human activity data

This section briefly describes the different types of features computed on each time series representation. The whole detailed list of all features is presented in Appendix A.

1.3.1 Features on signal, derivative and integral

The feature set computed on the signal, its derivative and its integral can be regrouped in two kinds : statistical features that are only based on time series values (independently to the time order) and temporal features that are strongly dependent to time series order. The first group contains for example max/min values, mean, variance and higher moments of the time series and the second group contains the peak number and how much time a threshold is crossed for instance.

1.3.2 Features on autocorrelation

The feature set computed on the autocorrelation relies on local minimums and maximums and their position on the time delay axis. As the notion of local extremum depends on the resolution of autocorrelation computation, a parameter is introduced corresponding to the minimum time distance between two extremum.

The amount of such local extremum corresponds to a first feature. Then from the list of local maximum the mean and standard deviation of their values are computed, as well as the mean and standard deviation of distances between two successive maximum. The same is done for the local minimum list.

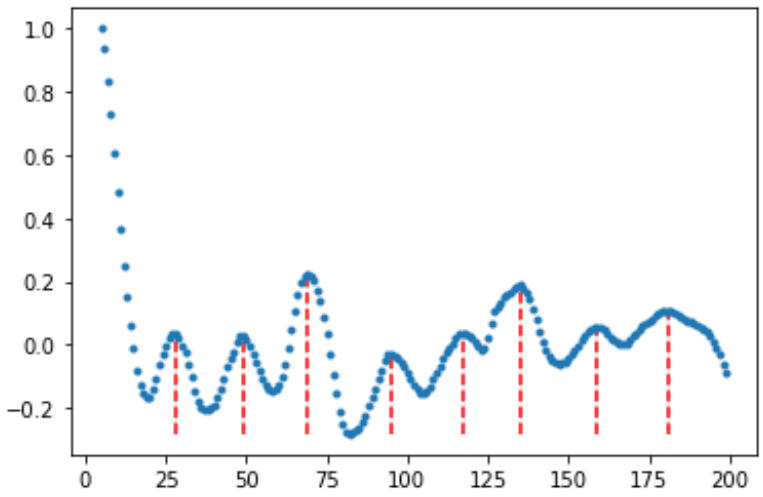


Figure 2.4: An example of feature on autocorrelation : mean distance between two successive local maximums.

1.3.3 Features on Fourier transform and spectrograms

The feature set computed on the Fourier transform is designed to capture the energy distribution with regards to different frequencies. The feature set computed on the spectrogram follows the same idea as for the Fourier transform with the difference that the time evolution of the spectral energy distribution is observable. For that purpose, several gaussian window filters are applied to weight a specific area of the spectrogram. To not introduce too much parameters in the feature design, these gaussian window are centered in the barycenter of the spectrogram energy. The window with an angle of $\theta = \frac{\pi}{4}$ emphasizes low frequencies at the beginning of the event and higher ones for the following time steps, and inversely for $\theta = 3\frac{\pi}{4}$. $\theta = 0$ focuses on one main frequency and $\theta = \frac{\pi}{2}$ on one time step which correspond to the coordinates of the barycenter. Moreover, the spectral and temporal widths of the part of spectrogram higher than a certain threshold in terms of energy also serve as features.

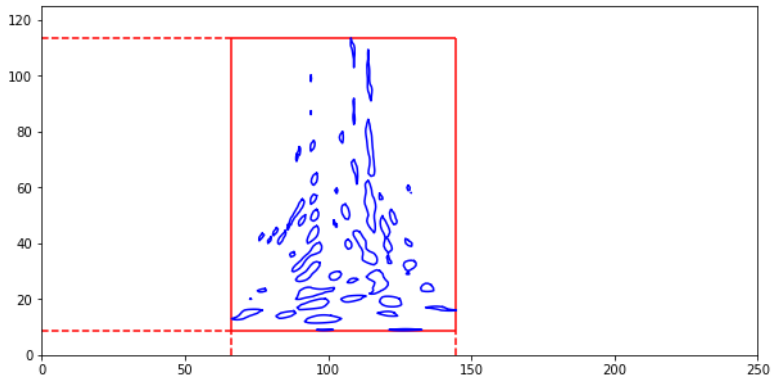


Figure 2.5: An example of feature on spectrogram : dimensions of the time/frequency window above a certain level of energy.

2 Supervised learning for human activity recognition

As described in 5.1 Tarkett aims at performing with its smartfloor several detection tasks that can be useful to monitor elderly patients health. For this purpose various experimental databases have been created, firstly to verify which detection tasks were conceivable with the technology and then to try to apply them in real conditions. These databases are composed by signals according to the targeted application with labels in a discrete space $\mathcal{Y} = \{1,...,K\}$ (with K the number of labels). As explained in previous section, from these signals are extracted features vectors of a space $\mathcal{X} \subset \mathbb{R}^p$ (with p the number of features). Statistical predictive models are trained with m inputs $(x_i, y_i)_{1 \leq i \leq m} \in (\mathcal{X} \times \mathcal{Y})^m$ to "map" a predictor function $\hat{h}$ from $\mathcal{X}$ to $\mathcal{Y}$ (a.k.a classifier if $\mathcal{Y}$ is discrete and regressor if it is continuous), usually by minimizing the *empirical error* relatively to a *loss function* l defined as :

$$\hat{R}_m(h) = \frac{1}{m} \sum_{i=1}^{m} l(h(x_i), y_i) \tag{2.1}$$

This approach is known as Supervised Learning ("supervised" because of the knowledge of labels for the training) and is justified by PAC theory proving that, under the assumption that (x_i, y_i) are drawn i.i.d from a probability distribution P over $\mathcal{X} \times \mathcal{Y}$, this *empirical error* tends asymptotically towards the theoretic *expected error* :

$$R_P(h) = \mathbb{E}_{(x,y)\sim P}\, (l(h(x), y) \tag{2.2}$$

After providing a short overview of supervised methods used in human activity monitoring, this section focuses on one particular supervised learning algorithm favored by Tarkett, namely Random Forests. It explains this choice and presents various results obtained with these models on different detection problems related to human activity monitoring.

2.1 Supervised models for activity monitoring : overview

Sensors and applications oriented overviews of human monitoring are presented in Section 1, as well as one about time series representation and features in Section 1.1. In the following are described several supervised learning methods used in related works about human activity monitoring.

Supervised learning models for human activity monitoring depend both on kind of signals and the detection tasks. As illustrated in Figure 1.1 the activity recognition task can be separated between short duration activities corresponding to a few seconds and long duration ones that can last several hours. Data representation and features extraction varies depending on this aspect and so is for predictive models. Moreover some applications attempt to discriminate one event from all the rest, like fall detection, whereas others aim at classifying multiple events or to achieve physical parameter estimation, like gait speed and strive. What also matters while choosing an appropriate model is the data nature and how much it is interpretable with regards to the prediction task. For example physiological parameters monitoring applications often come with body sensors providing signals close to the parameter to estimate, as described in Section 1.2 of Chapter 1.

Motion features estimation for short duration activities : While signals quality and accuracy are high enough to allow extraction of relevant physical features then predictive models of low complexity can be sufficient to get good prediction performance on short duration human activities. For instance fall detection can be achieved using only threshold-based classifiers with some wearable sensor like gyroscopes [29] or accelerometers [30].

Support vector machine (SVM) classifiers seems well suited for human activity detection based on motion features, especially using radial basis functions (RBF), because events are then characterized by the amplitude range of a few values describing motion physics. For example SVM are used for fall detection in [80] with accelerometer signals and in [273] for posture recognition with cameras. With accelerometers data, SVM have also been used in combination with auto-regressive models parameter estimation for activity recognition [118, 119]. Without wearable sensors, this kind of classifier can also serve to detect precise human motion, like in [8] for gesture recognition using cameras or for gait recognition using Wifi signals in [250].

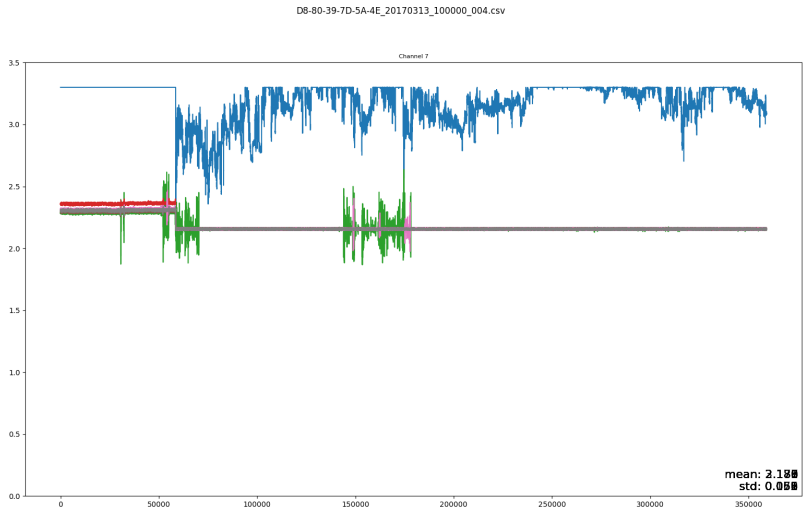


Figure 2.6: Example of the saturation effect on one channel of the electronic box. This can be caused either by a sensor default in one of the areas, or an electronic problem at wire levels or inside the electronic box.

In the case of Tarkett monitoring system threshold-based decision rules are used for basic routine work of the system including malfunction detection, activity detection (in the sense of non-silent signal) and entry/exit detection. Possible malfunctions are essentially drifts of the signal mean or brutal signal jumps (see Figure 2.6) due to technical reasons evoked in section 3.4 of Chapter 1. They are important to detect in order to avoid false alarms and other detection drawbacks, to help to fasten maintenance intervention and are easily detectable with a threshold on the mean signal value feature. Activity detection in each area of the installation is triggered by a threshold test on the mean energy of the signal fixed on a value supposed to be just a bit higher than noise mean energy. Considering entry/exit detection, it is handled by an intuitive handmade graph based model combining areas activation (especially the small entry area as described in Figure 1.7) and expert knowledge about the health institution schedule (like common lunch time), depending on hours these activations occur.

Time/frequency representations for short duration activities : While dealing with short duration events to detect using good resolution signals, pattern recognition methods can also be efficient to detect human activities. For that purpose existing works propose to directly segment times series of multiple sensors into core patterns through dictionary learning to help activity recognition [68, 159, 173]. When input data are spectrogram-like representations of signals, neural networks can also achieve pattern recognition in a similar manner than images classification [220, 270]. But more generally deep learning for human activity recognition is also usable on a feature space designed "by hand" with good performance [113, 161].

Despite these kind of models require an important computational investment, they present some clear advantages like their scalability to complex data distributions, their ease of implementation on raw data without much feature engineering and their ability to generate new samples from training knowledge.

Decision tree based models on time series : The random forest is an ensemble classifier invented about two decades ago [35], composed by decision trees. The decision functions of decision trees (from $\mathcal{X}$ to $\mathcal{Y}$) can be expressed as :

$$d_T(x) = \sum_{l \in \mathcal{L}} v_l \mathbb{1}_{[x \in l]} \quad , \tag{2.3}$$

with $x \in \mathcal{X}$ an element of the feature space and $v_l \in \mathcal{Y}$ the predicted classification or regression value . The set $\mathcal{L}$ corresponds to an ensemble of *leaves*, which are subspaces of $\mathcal{X}$ of the form : $\bigcap \mathbb{1}_{[\tau_1 < x_i < \tau_2]}$.

So a decision tree (DT) provides a partition of the feature space $\mathcal{X}$ labeled by values of $\mathcal{Y}$ and where borders of each partition subspace are orthogonal to feature variable axis. A random forest (RF) model $\mathcal{M}$ is then an ensemble of several DT $(T_1, ..., T_S)$ and its decision function is obtained by a majority vote between these DT for a classification or a mean prediction in a regression situation, which might also be weighted :

- Classification random forest decision function : $d_{\mathcal{M}}(x) = \underset{y \in \mathcal{Y}}{\operatorname{argmax}} \{ \sum_{i=1}^{S} w_i \mathbb{1}_{[d_{T_i}(x) = y]} \}$
- Regression random forest decision function : $d_{\mathcal{M}}(x) = \frac{1}{S} \sum_{i=1}^{S} w_i d_{T_i}(x)$

This formulation is common for any ensemble predictor [114], with S the number of sub-predictors in the ensemble and $w_i \neq 1$ if sub-predictors response is weighted. It is interesting to note that by superposing several labeled partitions corresponding to several DT of a RF, we obtain a new partition with borders that are still orthogonal to feature variables axis. This suggests that a RF classifier could be expressed as a unique decision tree (more details in Chapter 4).

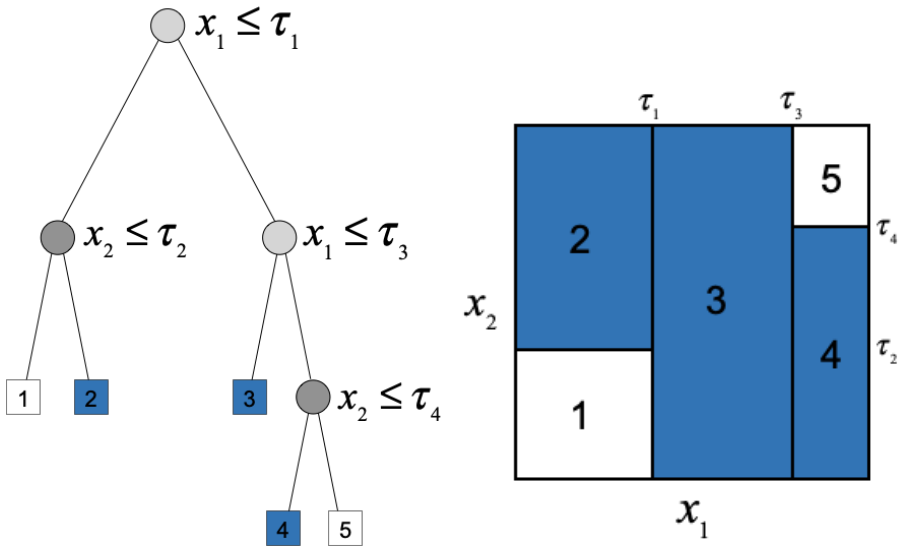


Figure 2.7: Example of a 2-d decision tree and its corresponding partitioning.

Random forest training algorithm: The training principle of random forests is also generalized to other ensemble predictors by the procedure known as "*bagging*" (for bootstrap-aggregating). Each DT of the RF is trained on a bootstrap version of the training dataset to reduce correlation between trees by the following process, which is known as CART algorithm. Given some bootstrap labeled samples, each tree is built recursively from the root to the leaves by applying the same splitting procedure for each node relatively to the part of samples reaching the node. While on a node n, this procedure looks for a pair of feature variable and threshold (ϕ_n, τ_n) that splits the best samples reaching this node, with the splitting efficiency based on an *impurity* measure. Impurity measures, which can also be found under the name of entropy, are used to quantify either uncertainty while dealing with a random variable or purity with regard to labels while dealing with a set of samples. The author of [64] describes a general entropy formula with a parameter β adjusting the concavity of the function and where $P = (p_1, ..., p_K)$ the probabilities for each label in case of a random variable and the label proportions for a set of samples :

$$H_\beta(P) = \frac{2^{\beta-1}}{2^{\beta-1}-1}\left(1 - \sum_{i=1}^{K} p_i^\beta\right). \tag{2.4}$$

This work also shows that this entropy is the same as the Gini index for $\beta = 2$ and the Shannon entropy while $\beta \to 1$, which are the two most common impurity measures for building decision trees. In this work we used the classic Gini index (usually preferred for computation simplicity) then defined as :

$$Gini(P) = 1 - \sum_{i=1}^{K} p_i^2. \tag{2.5}$$

At a given node n, CART algorithm chooses the split that maximizes the *purity gain* relatively to this measure defined as :

$$G(\phi_n, \tau_n) = Gini(P_n) - \alpha_l Gini(P_l) - \alpha_r Gini(P_r), \tag{2.6}$$

with P_n label proportions at the node n and P_l, P_r label proportions distributed on the left and right sides of the node n after the split, weighted by α_l, α_r, the part of samples reaching left and right sides ($\alpha_l = 1 - \alpha_r$).

It is also important to precise that for reducing further correlations between trees of the RF and to fasten the training, only a randomly drawn subset of all features are tested during the split selection, which is usually set to $\sqrt{p}$, with p the total amount of features [35].

Advantages of random forests : Random forests present numerous advantages as a supervised learning method but also with regards to industrial applications, which make them still largely used for regression and classification problems, despite their development occurred several decades ago. This model is very flexible as it has a scalable complexity through the amount of DT it encompasses and their depth, and it is then easily adaptable to the complexity of training data. Moreover, contrary to SVM or K-NN models for example, random forests are not very sensitive to distances between training samples in the feature space. Indeed as explained earlier, the impurity

criteria used to build DT node splits is independent to distances between samples and the split hyperplane, so this point makes this model very convenient while dealing with high dimensional feature spaces. The node splitting procedure also provides an inherent feature selection ensuring to ignore features that are useless for separating training data. Concurrently to this feature selection is provided a *feature importance* scoring, defined and exploited in Section 2.5, that allows better interpretation of the model and the training data.

More generally random forests come with several interpretation capacities which are highly valuable for industrial applications of machine learning. Added to this feature importance provided by this model, it can also be pointed out that decision rules of decision trees are by definition very interpretable compared to some "black-box" models such as neural networks. Finally the random forest training algorithm also allows to achieve generalization error estimation through "*Out-of-Bag*" scores (computed on samples that are out of the bootstrap sample set). It also permits fine clustering using data distribution in leaves partition [194]. Considering very practical real world aspects of models implementation decision trees represent one of the easiest model to implement without any statistical knowledge for developer teams and provide relatively fast predictions.

For all these reasons and encouraged by preliminary works on machine learning models comparison [172, 217] Tarkett decided to focus on random forests for the main part of their machine learning applications. What is shown in the last two chapters of this work is that decision trees can also be considered advantageous insofar as intuitive model updates are achievable in the context of model-based *transfer learning* and *budget learning*.

Random forests on TS for human activity monitoring : All these mentioned advantages of decision trees and random forests make these models good candidates for industrial applications of supervised learning, especially for supervised detection based on time series as for Tarkett monitoring system. Numerous works treat the problem of online learning of decision trees while confronted to real application time series. For instance [79] introduces the notion of "Hoeffding trees" using Hoeffding bounds to determine the amount of sample from which decision tree models need to be updated on data streams. Another work on decision trees learning from data streams [212] combines a variant of usual RF known as *extremely randomized trees* [99] and online bagging [188] to develop a new method a online decision tree learning. Some works study the question of using directly time sequences in node splits [74, 266], by modifying unique value threshold tests with TS similarity measures whereas some others induce temporality in tree models through, for example, the notion of "Markovian decision trees" [130, 137].

Another research topic about decision tree based models on real application time series concerns the adaptation of models while confronted with evolving data streams. Several works treat this question [94, 102, 167] and authors in [155] considers, on top of that, uncertainty on attributes. So the literature about decision tree based models on time series is very rich and often deals with very practical but also crucial considerations for industrial applications. Moreover the majority of previously mentioned works points out special concerns about the computational efficiency of these models, a question briefly evoked at the end of this chapter and more largely discussed in chapter 4.

Considering special applications to human activity monitoring, random forest models have been already proposed in the context of fall detection on Wifi signal [252] and RFID tag signals [15] and several comparative studies of supervised learning models include random forests, with interesting results for human activities recognition and with various sensors [6, 23, 76, 222, 272, 281].

2.2 Random forests on time series : application to fall detection

The first major application for Tarkett monitoring system is fall detection. Here is described the approach to build the initial fall detection algorithm based on a simulated database presented in Section 4.1 of Chapter 1.

2.2.1 Data processing

The experimental environment for this fall detection database is described in Section 4.1 of Chapter 1. Even though the experimental installation used for this database is composed by 3 areas (illustrated in Figure 1.7), each one was not plugged to the same input in the electronic box, so to take into account all the 8 input channels they are summed into one overall signal. After summing the different channels the signal is centered by removing from it a sliding mean and the noise is filtered by a second order butterworth low-pass filter. To pass from a signal database into a feature instances database for the training phase, first a sliding window is applied to get fixed length signal portions of size T_s (here $T_s = 2.5s$).

To avoid any signal location bias in training, signal extraction positions are drawn randomly (completely randomly for non-falls and for fall signals between window locations containing the fall). A data augmentation step is also applied in order to limit over-fitting and to increase data diversity a bit by selecting $N_s = 5$ extraction locations in each signal. Then all signal representations and corresponding feature sets described in the previous section are computed on each extracted signal portion, resulting in labeled feature instances database used for supervised learning.

2.2.2 Instantaneous fall detection

Every instance of the produced database corresponds to the whole feature set computed on a fixed size portion of signal. Training process is realized on these instances whereas the prediction is intended to be done on entire signal through a sliding window. A random forest model is then trained on the dataset composed by these instances and performance evaluation process is done by k-fold validation with $k = 10$. To avoid any over-fitting bias due to data augmentation (implying that several instances come from the same signal and can be very close) the k-fold separation is done on instances batches corresponding to original signals. Thus two instances of the same signal can not be in the training set and testing set at the same time.

2.2.3 Macro-decision detection with sliding window and prediction buffer

In practice in real conditions there is no random portion of signal drawn to compute features, but rather a sliding window inducing feature computation each new time step with a corresponding real time random forest prediction at the sampling rate

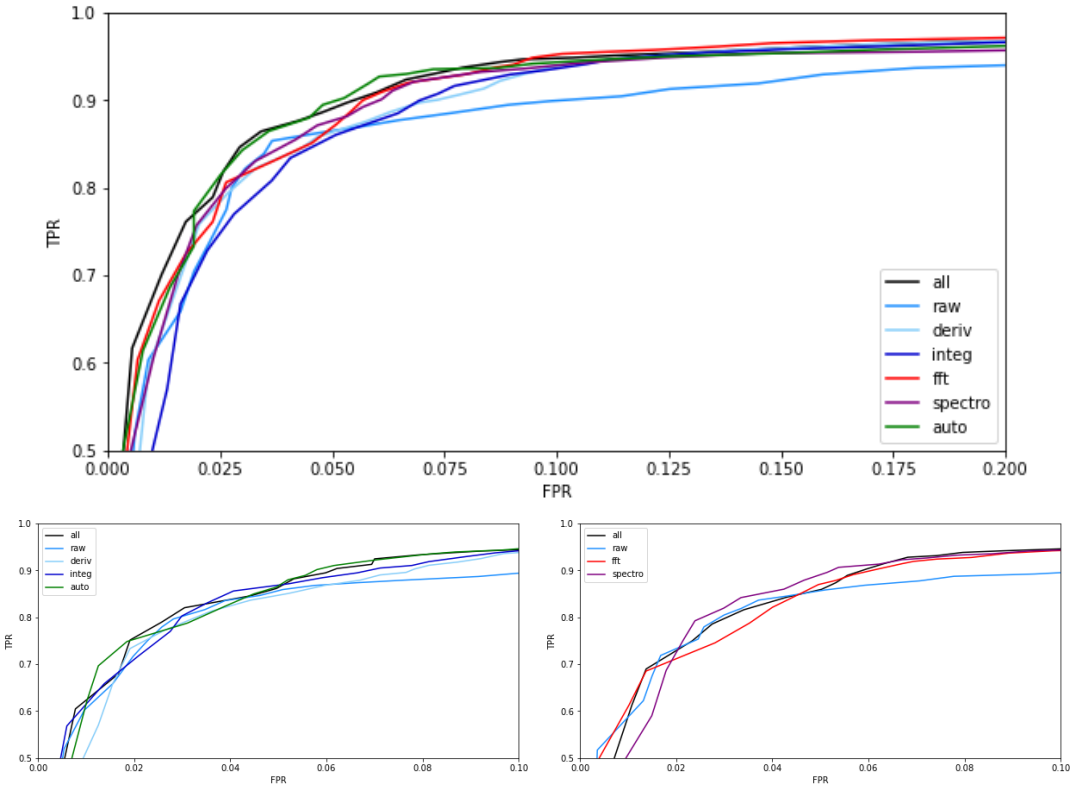


Figure 2.8: ROC curves on fall detection simulated DB according to each signal representation used to extract features.

$f_s = 100Hz$. The time evolution of this random forest response can also be useful to exploit. Indeed it is reasonable to at least assume that a fall duration is bounded in time, it can neither be too short (under 0.2 s) or too long (over 5 s).

For this reason a macro-decision model, based on the smoothing of random forest instantaneous predictions, have been developed. The final model decision $d_{\mathcal{M}}(s_t)$ of the random forest of size S, at time t on the signal value s_t, is based on the mean amount of trees predicting a fall during B previous timesteps, compared with a threshold τ_d:

$$d_{\mathcal{M}}(s_t) = 1 \Leftrightarrow \frac{1}{B}\frac{1}{S}\sum_{j=t-B}^{t}\sum_{i=1}^{S} d_{T_i}(x_j) > \tau_d, \tag{2.7}$$

with x_j the feature vector computed on the sliding window $[s_{j-T_s} : s_j]$.

As illustrated in Figure 2.9 the macro-decision allows to reduce the volatility of real-time random forest response, thus making the detection more reliable by avoiding some very short false alarms.

2.3 On-floor motion detection

The previous fall DB contains falls simulated by employees following a protocol that varies only in the fall direction : "Forward, Backward and Lateral". But one of the biggest challenge for efficient fall detection is "soft" fall detection, meaning while the

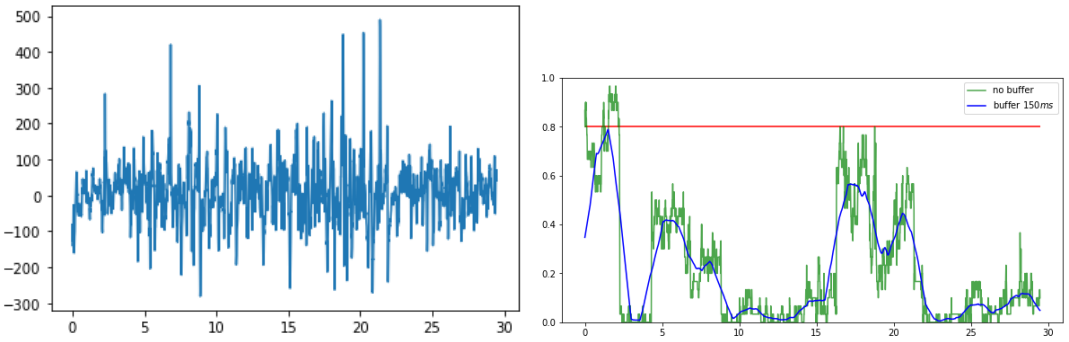


Figure 2.9: Effect of the macro-decision buffer on a real false alarm ($B = 0.5s, \tau_d = 0.8$).
Real example of a false alarm with the instantaneous random forest model. On the left is represented the original signal and on the right is represented the instantaneous random forest output (green), the smoothed version (blue) and the detection threshold (red).

fall is not brutal and the patient struggles some time before reaching the ground. In that case the patient is likely to be still conscious and moving on the ground. This is the motivation of this new detection task with this dedicated experimental dataset.

Furthermore signals of this DB are acquired by the Electrometer device able to measure very low charge signals and avoiding the use of an external amplifying circuit. This DB has been built concurrently to the design of ebox v2 to investigate if further applications of the floor sensor were achievable with better electronics, in particular exploiting low amplitude signals. So this experimental DB has been developed for two parallel research objectives : firstly to check if low activity signals were exploitable using our sensor and with an adequate electronic signal acquisition. In this case this study would be used to design a new electronic box able to deal with this kind of signals (ebox v2). Secondly the other objective was to study the feasibility of detecting, among these low activity signals, the ones corresponding to a person struggling on the floor, what might serve to enhance fall detection in "soft" fall situations. In this experimental context, this DB is annotated with numerous sub-labels that can be organized as follows (all details on labels are given in Table 2.1) :

- Walks : normal walks and simulated elderly walks.
- On-floor motion : including simulated events corresponding to what may happen after a fall if the faller remains conscious.
- Low amplitude signals corresponding to non-problematic activities : sitting on a sofa or a chair while balancing or doing feet movements.
- Undetectable events : silence or laying on bed, supposed to generate no signal on the sensor.

As data are obtained with a completely different device from ebox v1, their quality is highly better but signals are hardly comparable to real installed system signals.

2.3.1 Electrometer signals nature

Data processing is different here from the previous DB in the way that the signal does not need to be inverted (this step is due to the sign of the transfer function gain of the

Label	Description	Number	Macro-Label	Number
0	Walking (normal)	112	Walk	164
4	Walking (old)	52		
1	Sitting in sofa + feet movement	205	Move	417
2	Standing with balancing	212		
3	Silence	79	Low-signal	175
14	Laying on bed	96		
5	Complaining (back)	68	On-Floor	576
6	Complaining (side)	69		
7	Complaining (front)	16		
8	Crawling	62		
9	Trying to lean (wall/bed/sofa)	45		
10	Laying down (front)	1		
11	Laying down (back)	99		
12	Laying down (side)	85		
13	Leaned to a sofa (floor)	131		

Table 2.1: Description of the on-floor activity database obtained with the Electrometer.

external conditioning circuit, as expressed in Equation 1.5) and only one channel is recorded for each signal event.

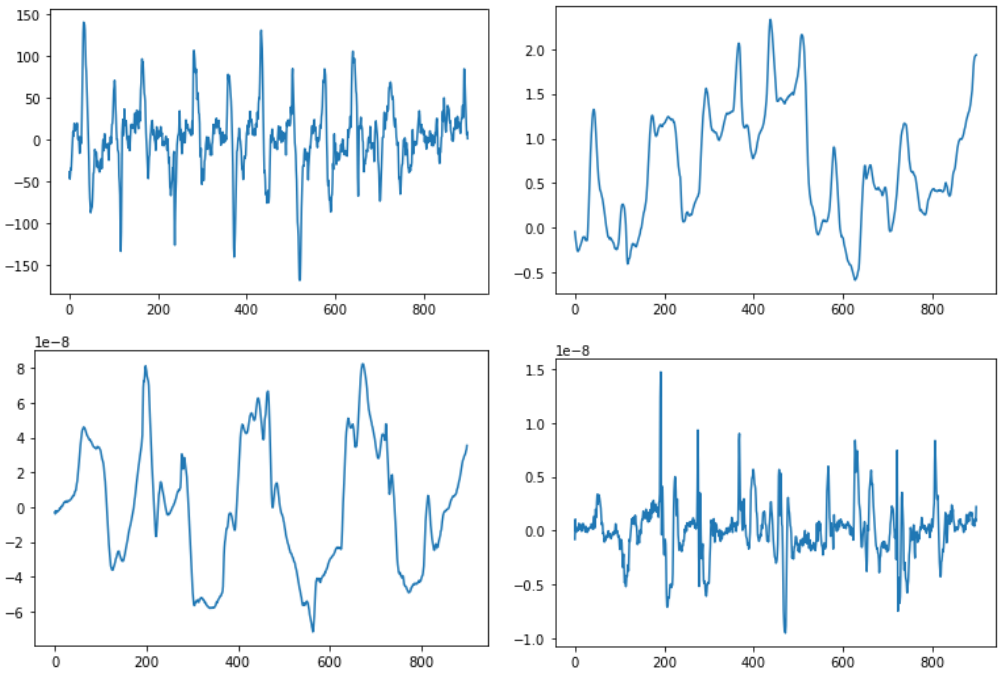


Figure 2.10: Comparison of the signal's nature between different electronics.
First row shows a walk signal acquired by **ebox v**1 on the left and its integral on the right, whereas second row shows a walk signal acquired by the **Electrometer** on the left and its derivative on the right.

As described in Equation 1.3, after signal conditioning circuit (for amplification purpose), the output signal is homogeneous to a current signal which is the derivative

of the original signal. This can be observed on Figure 2.10 representing walk signals with the two different electronic devices. So this relation implies further links between features and TS representations they belong to, while comparing the two databases. This suggests that a feature space harmonization, based on the theoretic electronic model knowledge, might be possible but it will not be discussed here.

2.3.2 Walk activity and low signals discrimination

The initial trained model on these data is a random forest classifier on three labels : *Walking*, *OnFloor* and *Other*. As shown in Figure 2.11, this models seems to provide drastically good classification for separating the walks from the rest of events, which is a promising result for extended elderly activity monitoring, as explained in details in the next application on walk recognition (see Section 2.4). On the contrary Figure 2.12 illustrates that distinguishing *OnFloor* movements from the rest seems far harder to achieve on these data.

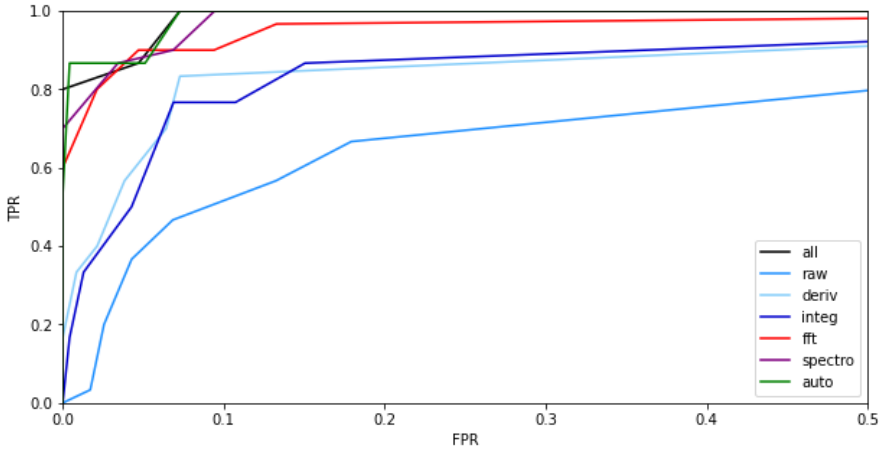


Figure 2.11: ROC curves of RF classifiers for *Walk* vs *Non-walk* events detection.

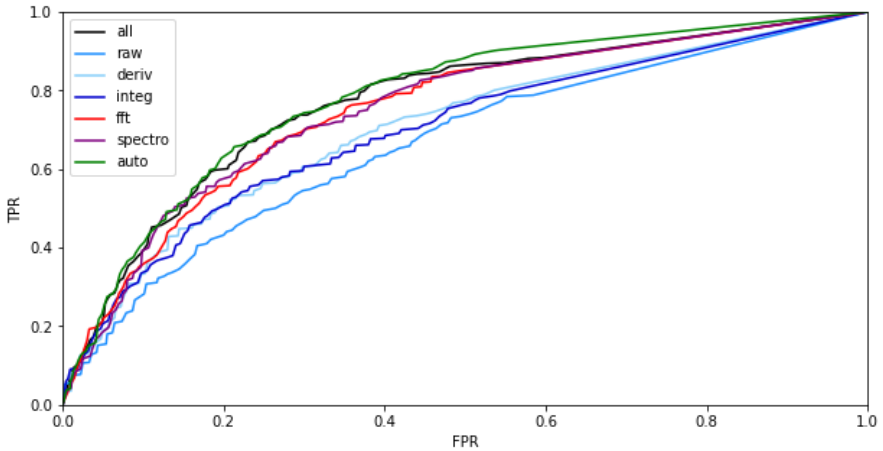


Figure 2.12: ROC curves of RF classifiers for *OnFloor* motion detection.

Nevertheless these observations have to be put in perspective for the following reasons : firstly as this DB is focused on low amplitude signal recognition it is biased such as the Walk is almost separable uniquely by observing the mean raw amplitude. Secondly as explained previously, the *OnFloor* event detection is intend to be used in combination with fall detection, so it would not be triggered only by itself. Thus even a small ROC area under curve score above 0.5 can be considered as promising for investigating strategies to enhance fall detection results.

2.3.3 Cascade random forests classifier

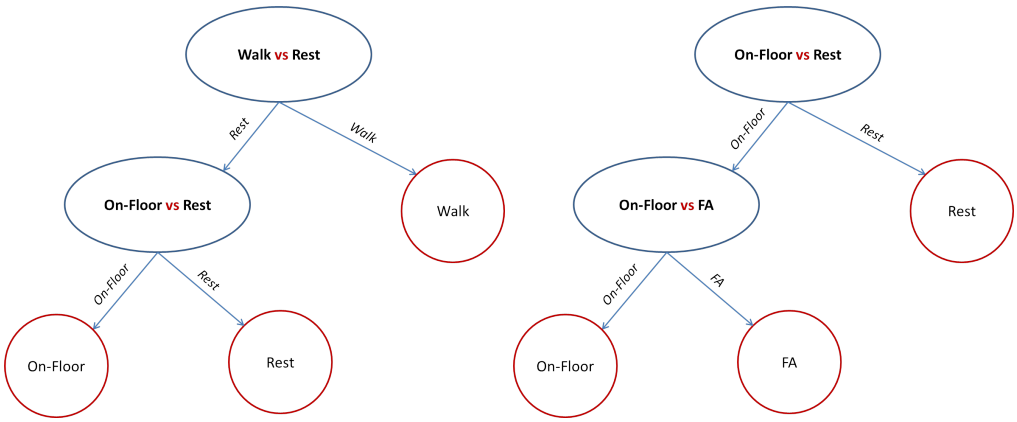


Figure 2.13: Illustration of the cascade classifier for *Walk* and *OnFloor* events detection.

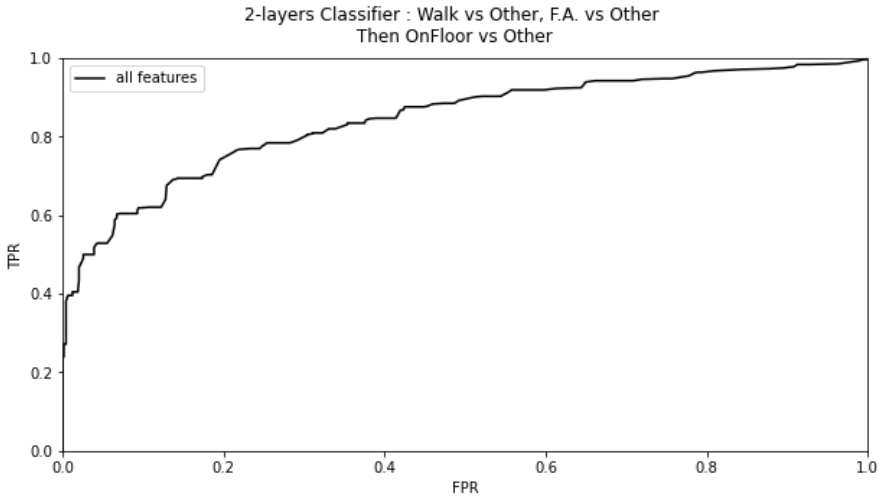


Figure 2.14: ROC curve of combined cascade classifier for *OnFloor* motion detection.

This model design is an illustration of the need of model interpretation : the final classifier is decomposed into three separated classifiers on different class sets to better understand how data are distributed and what are the situations provoking classification issues. Indeed predictions analysis of the first trained model showed that there was almost no confusion between the *Walk* class and the *OnFloor* class but false alarms were mainly due to miss-classification related with the two low amplitude

signals sub-classes. So we developed a cascade random forest model composed by 3 binary RF classifiers trained on different labeling : 1) *Walk* vs *Non-walk*, 2) *Non-walk* vs *OnFloor* motion and other low amplitude signals and 3) *OnFloor* vs false alarms among other low amplitude signals.

This combined model presents several advantages : it provides a refined multi-class classification and is more interpretable than a direct multi-class unique random forest. Moreover it presents better results with regards to the main interest label (*OnFloor* motion) as shown in Table 2.2.

2.4 Real walks recognition

Numerous medical researches about elderly health mention that gait shape can reveal important indicators about physical state degradation [115, 185], patient motion capacities [37, 180], the falling risk [87, 116, 210] and even mood disorders [73]. Moreover fall detection or these kind of dangerous anomaly alarms need to be triggered mainly while the patient is alone. Thus recognizing whether a patient is isolated or not may be a significant improvement to lower the false alarm rate for these applications. It may also serve to detect patient room night intrusions in elderly institutions.

As described in Section 4.2 signals of this DB come from a walkway in an elderly healthcare institution, composed by more than 10 consecutive small areas of sensor, connected with the same signal conditioning electronic circuit than ebox v1. So the difference in data processing for this DB is mainly the number and the size of areas, each providing a signal, which allows to exploit spatial information along the pathway.

Patient walk recognition

This DB contains a double labeling focused on walk situation recognition, one label indicates if the signal corresponds exclusively to elderly people on the walkway, to medical staff people or if it is both. The other label indicates if it is a walk without any equipment, if a cart or a wheelchair is used, or if it is a mix walk (often the case while someone is pushing a wheelchair of someone else).

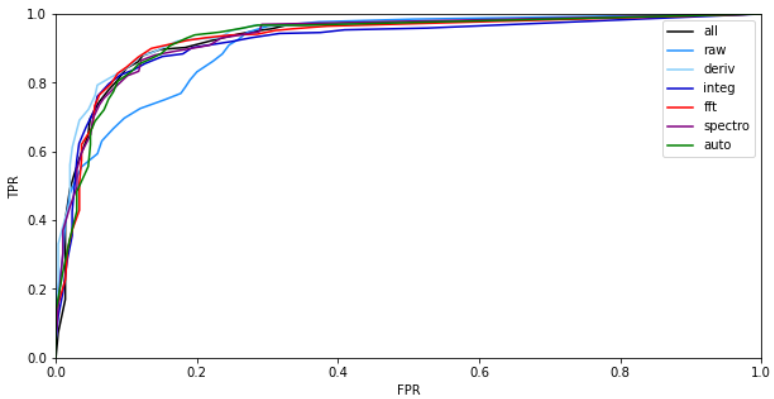


Figure 2.15: ROC curves of elderly walk recognition task on the corridor.

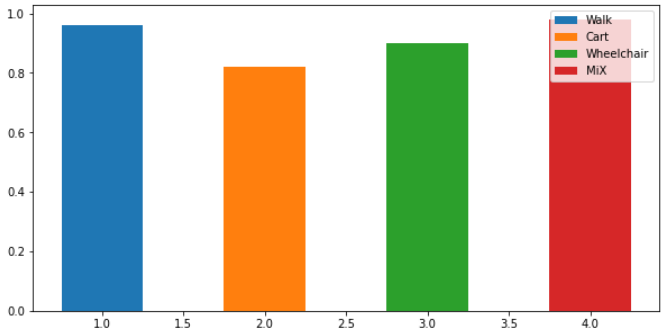


Figure 2.16: One vs all mean AUC of ROC curves on the walk equipment recognition task.

Gait features

In order to quantify gait frequency and stride speed we consider only one-person walks on which are computed two values : the mean distance between local maximums of autocorrelation and the inverse of mean time passed on each area of the walkway (that are supposed to be of the same length) during the walk. These can be considered intuitively as good approximations of *step mean duration*, which is the inverse of *gait frequency*, and *stride speed*.

Such gait features are known to be important indicators for patient global health state assessment and are even correlated to fall risk. A close view on these values over the *Real Walk* database presented in Figure 2.17 confirms the interest of autocorrelation representation for the walk event. It also shows that elderly people can be distinguished from non-elderly ones, which can have interesting application for activity monitoring in elderly healthcare institutions, especially for separating staff activity from patient activity.

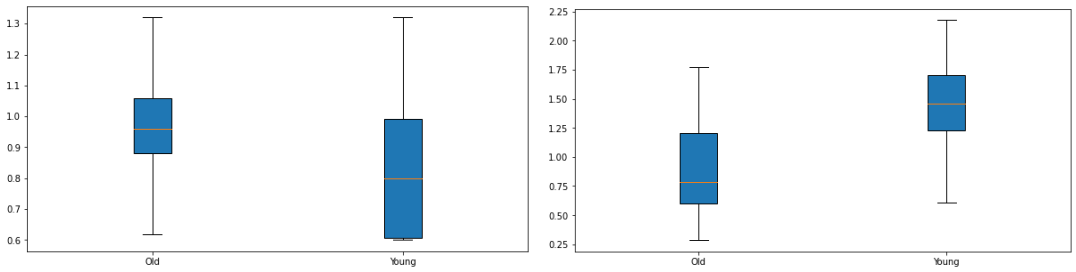


Figure 2.17: Gait features comparison between elderly and non-elderly people.
Left boxplot represents the mean distance between local maximums of autocorrelation, it is part of the total feature set presented earlier. The right one represents the inverse of the mean time passed on each area of the walkway while walking on it. It is proportional to the mean walking speed.

Practical limits in daily life monitoring

Beside the interesting properties of the mentioned gait features and the achievable detection tasks presented on this DB, these observations remain yet experimental for two main reasons. Firstly the data collecting context is different from patient room

installations as distances between room extremities are a lot shorter which is not enough to build stable gait indicators and sensor areas configuration does not allow to extract any spatial information contrary to the walkway installation. Secondly, this DB labeling was possible only through cameras installed in this public location with the agreement of resident and workers but is much harder in personal resident rooms for obvious confidentiality reasons. Then a major practical obstacle remains to confirm and extend this experimental study into elderly room monitoring installations.

2.5 Conclusion

In previous sections is presented the ensemble of TS representations and features extracted we built to obtain our datasets on three key health monitoring applications : fall detection, on-floor motion detection and walk analysis. These datasets allowed us to train several models based on random forests that show promising results on these experimental data like around 90%,5% of TPR/FPR for fall detection, a combined classifier able to detect motion of people on floor with more than 80% of ROC AUC score and models that are able to distinguish efficiently elderly and non-elderly walks as well as walk equipment use.

Each of these models are still considered experimental because of at least one major difference with real conditions : the fall detection DB is composed by simulated falls and non-elderly people, the on-floor motion DB is gathered by a particular electronic acquisition device of high quality with different signal's nature, the real walk DB is obtained this time in a real healthcare institutions but in a long walkway with several sensor areas whereas Tarkett's monitoring system is installed in patient rooms with poor spatial information, which excludes for now some of the gait analysis capabilities.

Performances on experimental conditions

DB	Falls	Falls(M)	On-floor	On-floor	On-floor(C)	Real Walk	Real Walk
Label	Fall	Fall	Walks	On-floor	Both	Elderly	Equipment
Raw	96.0	96.5	93.5	67.2	71.1	92.9	88.7
Derivative	97.1	97.3	98.5	71.1	72.0	94.5	92.9
Integral	97.3	97.5	99.1	71.0	65.4	94.1	92.7
FFT	97.2	97.0	99.9	77.0	71.3	94.6	92.1
Spectro.	97.6	97.8	99.9	77.3	69.5	94.6	91.6
Autocorr.	97.5	96.3	99.9	78.9	65.5	94.0	93.2
All	97.4	97.5	99.8	78.8	84.3	94.0	92.3

Table 2.2: Mean ROC AUC by feature group and classification task.
The second column refers to fall detection with macro-decision described in Section 2.2.3. On-floor(C) refers to the 3-layers combined classifier described in Section 2.3.3. The last column is a mean AUC score over several one versus all classifications based on walks labeled by equipment use.

So the results on these applications mainly serve to study what seems achievable with the monitoring system, even if some of these application need technical enhancement (for instance in electronics or installations) to be usable in real conditions. Summarized results are presented in Table 2.2, they have also been assessed

separately with random forests trained on each TS representation in order to observe their relevance with regard to each application and labeling.

In the majority of detection tasks it seems that it is not necessarily better to use features of all time series representations. On the contrary, using only features of the original raw signal itself clearly shows lower performances. This suggests that a unique well suited representation for each detection task should be sufficient. We can also remark that some representations, like spectrograms or autocorrelation, regroup less features than other ones and are however well performing from a prediction accuracy perspective. Nevertheless even if the mean ROC AUC of models built on each group of features are comparable, one may prefer one representation based on TPR/FPR trade-offs reached by each ROC curve.

Importance of TS representations and features

The feature importance provided by the model for a given feature corresponds to the cumulated impurity decrease in decision trees splits, weighted by the amount of samples reaching these splits. Relatively to a decision tree T the feature importance of a feature ϕ is :

$$Imp(\phi) = \sum_{n \in T} \mathbb{1}_{[\phi_n = \phi]} p_n \Delta G(\phi_n, \tau_n) \quad , \tag{2.8}$$

with $\Delta G(\phi_n, \tau_n)$ the Gini gain at node n as defined earlier and p_n the proportion of samples reaching this node. Then the total feature importance relatively to the random forest is obtained by averaging feature importance over all the trees of the RF.

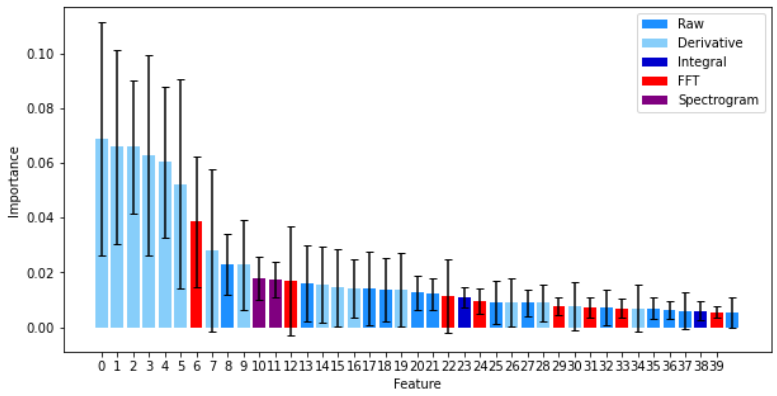


Figure 2.18: Feature importance of the 40 best features on simulated falls DB by signal representation.

As it is shown in Figure 2.18 main features chosen by the random forest model for the fall detection task belong to the derivative representation of the signal but some features from raw signal, Fourier transform and spectrogram are also impacting. Another observation is that the standard deviation of this feature importance is non-negligible, which suggests some uncertainty about this ranking beyond the 10 first features. This can be explained by the inherent randomness in the feature selection process of random forests.

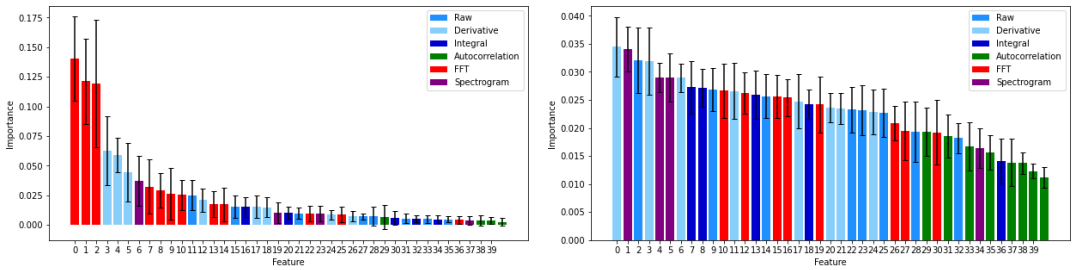


Figure 2.19: Feature importance on the *OnFloor* DB by signal representation and depending on labeling.
On the left is represented the features importance relatively the *Walk* detection task and the *OnFloor* detection task on the right.

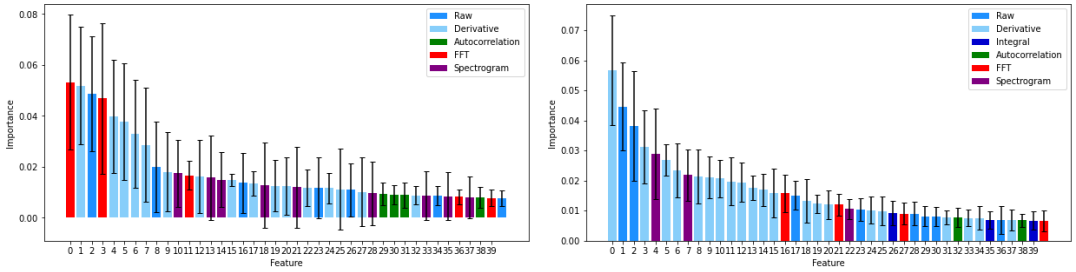


Figure 2.20: Feature importance on the *Real Walk* DB by signal representation and depending on the amount of features considered at each split.

Indeed Figure 2.20 shows that this high variance effect is a bit reduced while considering all p features in each split selection instead of $\sqrt{p}$ features. It is also observable in Figure 2.19 that, depending on the chosen labeling, features and TS representations ranking can vary a lot. This is also understandable in terms of mutual information, another feature importance measure discussed in the next section.

Even though the derivative representation is over represented while observing this feature importance ranking, the random forests have been also trained separately with features of each signal representation to observe their discriminatory potential regarding the fall detection task. Table 2.2 confirms that there is no real predominance of derivative representation in any of the considered detection tasks. Surprisingly it seems that it is possible to select indifferently any TS representation without big performance degradation, from the moment it is not directly the raw signal. Nevertheless by looking over all the detection tasks, TS representation encompassing frequency information (Fourier transform and spectrograms) seem a bit more reliable regarding mean performances.

3 Real constraints and interpretable sparse models

As explained the previous detection task results are experimental and ignore several constraints that need to be taken into account in real conditions. Indeed we evoked sensor installation constraints with a variable sensitivity and connectivity constraints, making hard to use a lot of areas and to obtain exploitable spatial information. Other obvious constraints concern data gathering and labeling on real environments. More-

over constraints on electronics and signal quality are neither negligible (real installation electronic suffers from noise and signal resolution is limited to 12 bits), which makes for example the on-floor motion detection not directly applicable.

In addition the different TS representations come from the same information contained in the original signal and most of the associated features are handmade. So the whole feature set presents undoubtedly some inner correlations. For these reasons our models need to be sparse in computed TS representations and features but also regarding the amount of used decision trees.

3.1 Time series representation complexity

The usefulness of each signal representation regarding prediction accuracy, discussed in previous section, is not the only criteria motivating its computation. Indeed firstly signal representations interpretability is important to consider, especially for experimental data exploration. Secondly the computational resources they require is also crucial to take into account for the monitoring application to work in real conditions. Detection tasks of Tarkett monitoring system have to be embedded in a small device with limited cpu resources of 256kB ROM, 16kB RAM and 40 MIPS. Ideally the embedded system must give a prediction on the current signal every 10 ms (same as sampling frequency), but all these features can not be computed with the device resources in this short time.

Signal representation complexity mainly depends on the size N of signal's window. If a random signal portion of size N is drawn without any prior knowledge, derivative and integral have linear computation complexity relatively to N, the fast Fourier transform algorithm is known to have a $Nlog(N)$ complexity, as well as autocorrelation computation. As previously explained, Wigner-Ville spectrogram can be obtained by computing N times fast Fourier transform on instantaneous autocorrelation at different time location, which results in a complexity of $N^2log(N)$.

3.2 Feature set inner correlations and mRMR criteria

Although correlations between features are not very problematic in an experimental context and might sometimes even help data analysis, it is an important concern when it comes to algorithm embedding as it is a source of unnecessary additional computations. There are mainly two causes of these correlations : those coming from the specific data distribution or inherent correlation in formal definitions of features themselves independently of data. Figure 2.21 represents correlation matrices estimated on each of the three datasets presented earlier in this chapter (left side shows absolute values of correlations and right side shows absolute values that are over 0.8). Even if these three datasets represent different kind of data (*OnFloor* DB is even different in the nature of signals it comes from), they show comparable blocks of highly correlated features. So this suggests that the source of feature set inner correlations is unlikely to be data distributions but rather feature definitions. It is understandable as both time series representations and some features extracted from them are relatively redundant versions of the same information.

In Section 2.5 the relevance of TS representations and features is studied for each classification problem through the feature importance metric given by random forest models. Additionally to high variance observed on those results for some classification

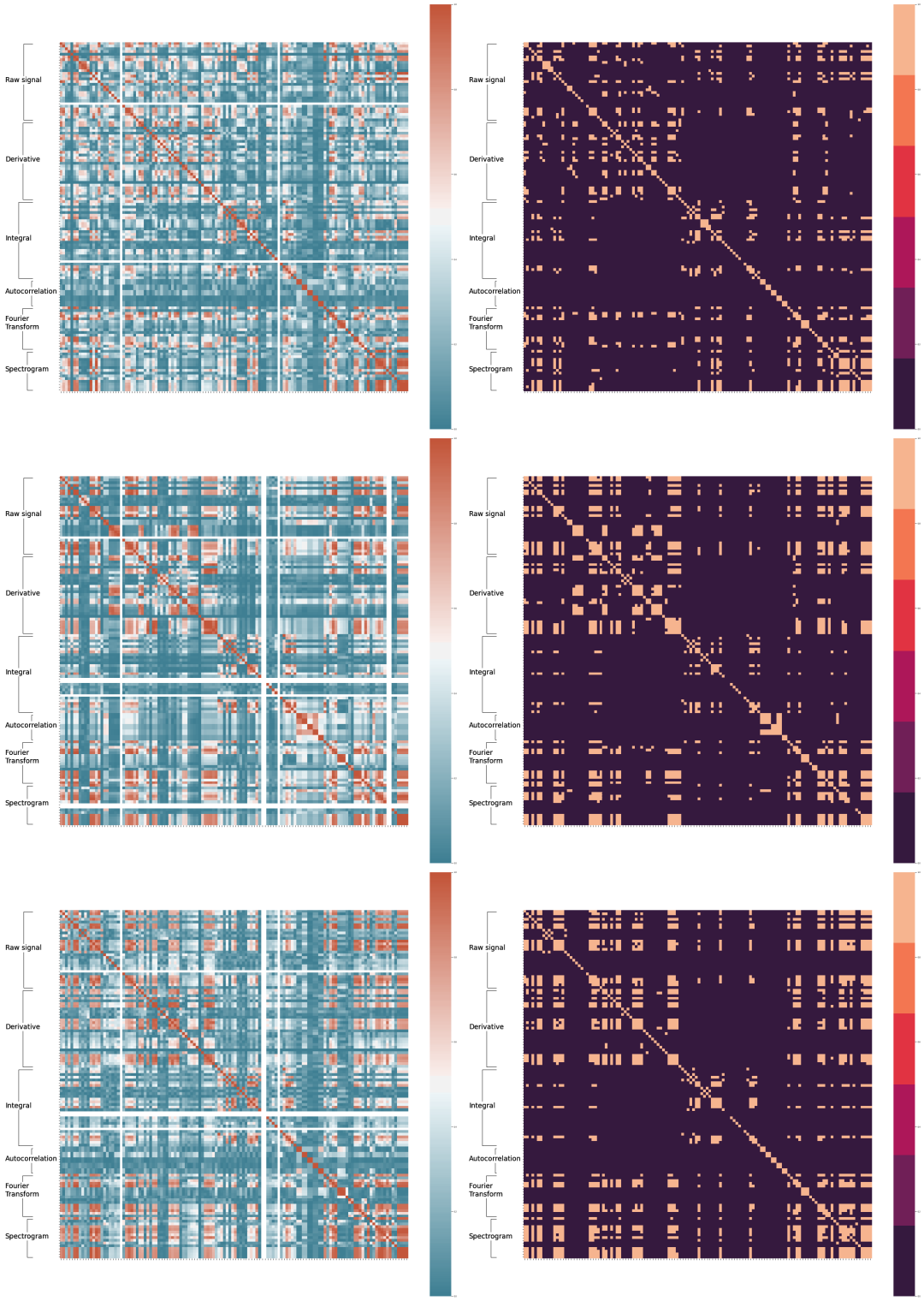


Figure 2.21: Correlation matrices on three datasets with features grouped by signal representation.

From top to the bottom : *Simulated Falls*, *OnFloor* and *Real Walk* databases.

problems (see Figure 2.18), there are two main drawbacks of feature importance measure : it is firstly model-dependent by definition and it also does not take into account feature correlations.

One model independent measure to assess the discriminant relevance of a feature is the *mutual information*, which itself comes from the notion of log-entropy (as it belongs to the same function family as Gini index [64]). Given two random variables (X,Y) following distribution P over $(\mathcal{X},\mathcal{Y})$, mutual information is defined as :

$$I(X,Y) = \mathbb{E}\left(log(\frac{P(X,Y)}{P(X)p(Y)})\right) \quad . \tag{2.9}$$

Mutual information between a feature variable and a label variable quantifies the efficiency of a feature variable at discriminating the label variable whereas mutual information of two feature variables is a measure of similarity/redundancy between them. Therefore it is a common tool for feature selection procedures in general [143, 147, 242], as well as for activity recognition applications [108, 159]. Based on this notion, authors in [77, 193] define the *minimum redundancy maximum relevance* criterion (mRMR), which is a measure that can be used to compare feature sets of same size both in terms of discriminant power and non-redundancy expressed as :

$$V_{mRMR}((x_1,..,x_K),y) = \sum_{i=1}^{K} I(x_i,y) - \sum_{i\neq j} I(x_i,x_j) \quad , \tag{2.10}$$

with $x_1,..,x_K$ random variables corresponding to K feature variables and y the class variable to classify. It also proposes a practical way to minimize the empirical approximation of this value for a given K, which is available online[1]. Even though the optimality and the stability of this feature ranking method can be questioned [203], it provides a simple way to get feature subsets both relevant and non-redundant, independently to the classification model.

3.3 Prediction similarities between random forest trees

While building a random forest each tree is trained using a randomly selected subset of features (usually of the size $\sqrt{p}$ with p features in total). This is meant, as bootstrapping training samples, to obtain a form of diversity within the ensemble of decision trees, by forcing them to use different features and be non-correlated. This notion of diversity between base classifier ensemble models is a key principle to explain the efficiency of majority vote [24]. Nevertheless while there are high correlation between features, feature subsets decision tree training may be not sufficient to create this ensemble diversity and we previously showed that our total feature set presents some correlations.

Indeed looking at detailed individual decision tree predictions we can observe that some groups of decision trees provide the same predictions on almost all testing data. Figure 2.23 represents these correlations between decision tree predictions computed on all test data for a fall detection random forest of size 10 . This can be due to either decision tree presenting close decision functions or to a lack of diversity in testing data themselves. This decision tree predictions correlation can be considered as a way to quantify "diversity", representing the inner classifier agreement inside the random

[1] http://www.home.penglab.com/proj/mRMR/

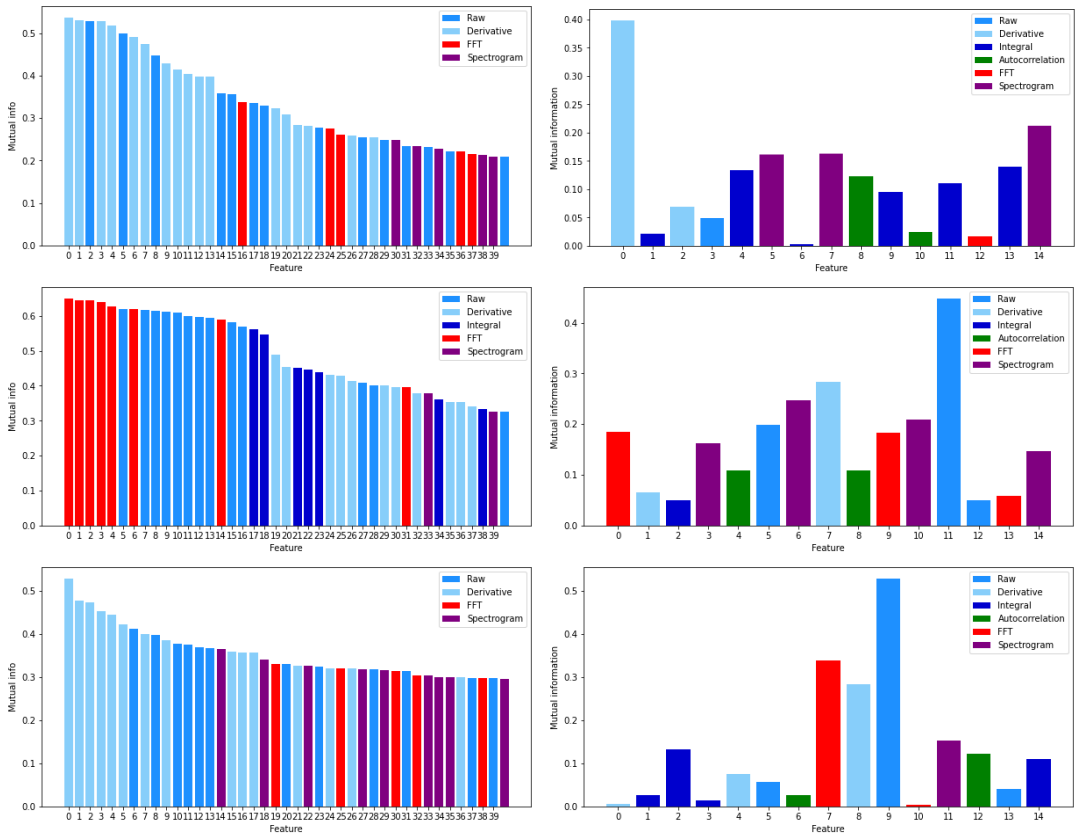


Figure 2.22: Mutual information between features and labels on three datasets (*Simulated Falls*, *OnFloor* and *Real Walk* databases) and feature sets of size 10 selected by mRMR criterion.

For computational reasons, we restricted the size of mRMR feature subset at 10. As explained in [193], the complexity of the used heuristic is $O(N.K)$ with K the number of features to select and N the number of discrete values applied on data.

forest. But there are plenty of measures used to quantify this notion under several names like "similarity, distance or ambiguity" between decision trees [146]. Moreover relationships between prediction accuracy and diversity is still a vast topic and an open question [24, 211].

Our observations may suggest that random forest size might be too much high but also that our final random forest models could be compressed into a smaller ensemble of decision trees. For example in order to avoid redundancy between decision trees authors in [136] developed "orthogonal decision trees" using Fourier representation of trees decision function and using orthogonality within the functional space. A recent algorithm addresses the problem of random forest compression in an extreme case by selecting a unique decision tree representative of the whole random forest [255]. So dealing with this prediction similarities between trees of the same random forest to get more compact prediction models would be an interesting perspective for our applications.

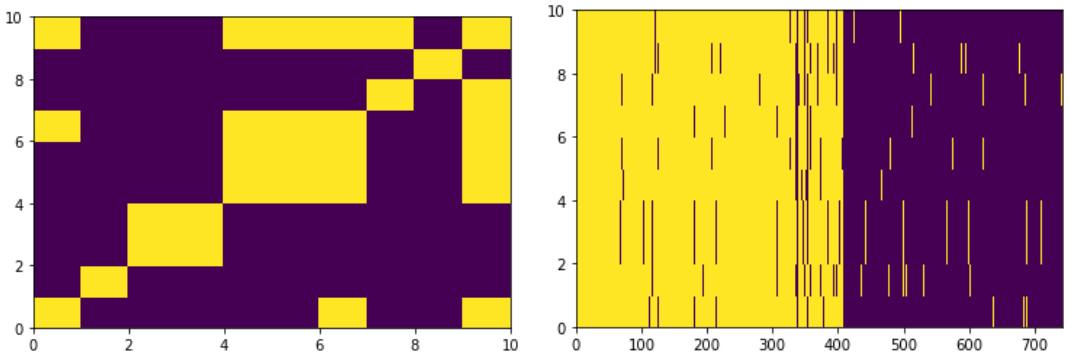


Figure 2.23: Illustration of the similarities between individual decision tree predictions of a size 10 random forest trained on simulated falls DB.
On the left the correlations between binary predictions of decision trees are computed on the whole data set. Yellow squares represent two decision trees having more than 0.9 correlation between their predictions. On the right is represented individual decision tree outputs over every instance of the sorted DB (falls from 1 to 409 and non-falls from 410 to 742).

4 Conclusion

Experiments presented in this chapter are promising on three main categories of detection tasks that are important for elderly monitoring : fall detection, walk recognition and on-floor movements detection. This advocates for the feasibility of a complete monitoring system based on plural signal representations and our wide feature set using decision tree based models. Moreover these experiments show that the relevance of each signal representation and feature varies according to the detection task and that additionally to random forest models, specific strategies can help to improve performance and interpretation. In particular smoothing real-time random forest response can be useful to reduce the false alarm rate of rare events and cascade classifiers are more interpretable and can be more efficient in a multiple labels classification.

These results are nevertheless still obtained in experimental conditions and are not directly exportable to real environments. Indeed they differ from real conditions in several ways : the data source, the type of sensor installation and the technology used to extract signals. This fall database is composed by simulated events by non-elderly people, walk recognition database comes from a walk-way installation (not patient rooms) with multiple sensors allowing to use spatial information (allowing to assess gait speed for example) and on-floor database signals have not been extracted with the same electronics as ebox v1 but the Electrometer instead, which a far more sensible and precise device (see Section 3.3 of Chapter 1).

Next chapters aim at tackling two main obstacles related to real conditions. Chapter 3 addresses the question of integrating new real data that are highly imbalanced into the previously trained experimental fall detection model using *transfer learning* methods. Chapter 4 deals with computation costs aspects for the embedding of real-time models. Indeed concerning mentioned feature redundancy or decision tree prediction similarities, random forest models are able to deal with these issues from a prediction accuracy perspective. Our experiments show that from a certain point, increasing the size of the features set or the amount of trees does not enhance the prediction accuracy [174] but it does not degrade it either, even with evoked forms of

redundancy. However, from computational effectiveness point of view, these aspects are crucial. Feature correlations and mRMR criterion, as well as prediction similarities, have been guidelines used by embedded system engineers to design the first fall detection embedded model. However computational costs can be taken into account in model design in a more general and adapted way. It is known as the *budgeted learning* framework and this is the topic of Chapter 4.

3

Transfer learning : from experimental setup to operational environment

1 Heterogeneous data generation

Previous chapter provides several examples of random forest classification models trained on different kinds of events : *falls*, *walks*, *on-floor motion* and other daily life events. These models on experimental database show the utility of Tarkett smart-floor in achieving elderly daily life monitoring but a limit is remaining : the lack of labeled real data. Indeed in order to label properly sensor signals cameras are used in parallel with the smart-floor, thus the only real environment data presented in previous chapter is the *Real Walks* database recorded in a nursing home pathway, which still differs from elderly private rooms data as explained in 2.4.

Based on feedbacks of the first fall detection model and with the joint work of medical staff a *Real Falls* database has been gathered without using any camera. After a few months this model allowed to extract a lot of false alarm events, some detected real falls and other undetected ones that were reported and then extracted and labeled by Tarkett approximately based on medical reports and signal observation. So the generation process of *Simulated Falls* and *Real Falls* databases differs on several aspects : values of sensor sensitivity and sensor area sizes, people executing activities and the events that are mainly represented, with an unavoidable change in the overall proportion of each events. For instance falls represent more than half of simulated events while they are representing less than 10% of events in the real one.

Thus with these new real data, questions arise about the need and usefulness of updating the fall detection model. We are in a situation where on one hand a *source* model, obtained with simulated data training, is largely implemented in elderly rooms equipped with Tarkett system. On the other hand, a real events dataset is gathered representing the *target* data on which we aim at obtaining the most accurate predictions. From that we can wonder if the *source* model is good enough on *target* data or if it is better to train a completely new model trained on these new data.

1.1 Fall databases comparison

Figure 3.1 compares mean ROC curves obtained with 5-folds cross-validation trained and tested on each domain. First *source* model seems as good on *source* domain as it is on *target* domain, which is comforting about representativeness of simulated events recorded in the first database. Secondly *target* model assessed on *target* domain shows the best performance which suggests that better prediction accuracy may be obtained exploiting *target* data. However a major issue is that *target* model performs particularly poorly on *source* data, because while considering a model update we expect a new model to be at least able to classify efficiently events of the simulated database. This might be due to a lack of variety in the real events database as it has been obtained by automatic data gathering whereas simulated data follow a precise generation protocol to be as diversified as possible.

Then we would like to enhance the previous model trained on simulated data by updating it using new real data but without loosing accuracy on original simulated data, as it is considered as our reference domain (because of precise labeling, contextual informations and corresponding videos). This is a typical model-based transfer learning situation where we aim at adapting a model already trained on a first domain using

data coming from another domain. These notions relative to *transfer learning* are detailed a bit later after basic databases comparison between simulated and real data.

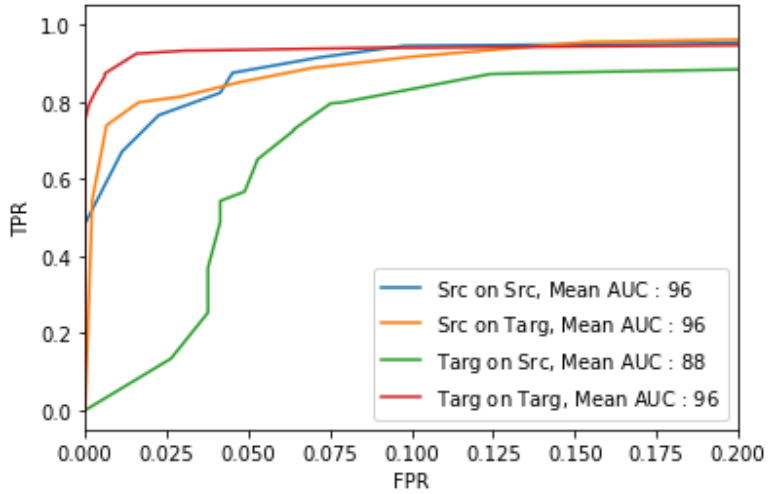


Figure 3.1: Fall detection datasets comparison based on ROC performance on each domain.

To ensure that transfer learning approach is suited for this situation it is necessary to verify we are actually confronted with different domains. For these purpose these two databases have been regrouped according to their label (falls and non-falls) for figuring out if real data were separable from simulated data. A simple random forest classifier of size 10 has been trained with our feature space for each event type (falls and non-falls) using labels that correspond to the database samples belong to. Figure 3.2 shows ROC curves obtained with OOB samples estimation, where AUC are clearly higher than 0.5 both for falls and non-falls, proving databases separability. Theoretic relationship between AUC optimization and non-parametric multivariate two-sample homogeneity testing problem is established in [55] and exploited using random forests in a comparable manner in [16].

Thus these observations confirm that these two databases come from different domains yet related, with possible enhancement of the *source* model using *target* data, justifying to investigate further in transfer learning capabilities. In this chapter we focus on transfer learning for fall detection application but as illustrated in Figure 3.3 several other use-cases could have been considered because a lot of different databases are available for different detection tasks (see Chapter 2) with heterogeneous data between them even though sharing some common events, each one gathered with a particular technology on a specific environment. The methodology presented here to compare databases and furthermore to perform transfer learning for fall detection can be extended in the future to any other detection task

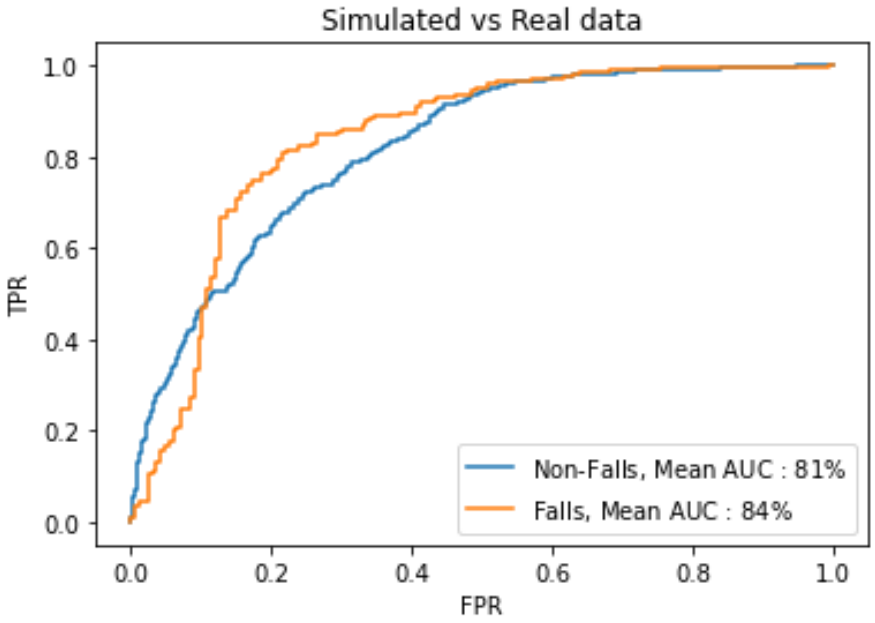


Figure 3.2: Simulated/real fall detection databases comparison for fall and non-fall events.

2 Overview of transfer learning

As mentioned in previous sections, the preliminary application of Tarkett which is also the core of their monitoring system to enhance elderly people living condition is the fall detection. For obvious reasons Tarkett could not wait to gather real fall data to develop this system, that is why a first model, presented in 2.2, has been trained on a simulated fall database and a simplified version of this model fitting embedded system resource constraints has been implemented and largely deployed in real installation in 2016. The first year the main real ground feedback was a high false alarm rate reported by medical staff although some real falls remained undetected (not the majority). In parallel other experimental studies have been conducted to improve the whole monitoring system (as those presented in previous chapter) and real data gathering and labeling allowed to build another fall database (described in section 4.3 of Chapter 1), this time composed by real fall and non-fall events, available in 2018.

Based on these observations it seems that these two DB present an important dissimilarity while still being related. Apparently re-train from scratch a new model with the new DB seems better than applying the one trained on simulated fall DB on real data, but we can wonder if the first DB and models are now useless from this point or if there is a way to benefit from them as they are related. This is the kind of question raised since the past few years by a growing field of interest under the general name of *Transfer Learning* which regroup numerous research sub-fields such as sample selection, covariate shift or domain adaptation. In other words, transfer learning aims at learning *task* that benefits from knowledge of different *domains*, where these two core notions can be defined as follows :

Definition 3.1 (Domain)**.** *A domain $\mathcal{D}$ is defined by two parts, a feature space denoted $\mathcal{X}$ and*

a marginal distribution $P(X)$ over this feature space. We denote $\mathcal{D} = \{\mathcal{X}, P(X)\}$.

Definition 3.2 (Task). *Given a domain $\mathcal{D} = \{\mathcal{X}, P(X)\}$, a task $\mathcal{T}$ is then also defined by two parts, a label space denoted $\mathcal{Y}$ and a conditional distribution $P(Y/X)$ relative to the feature vector variable. The unknown Bayes classifier is defined on $x \in \mathcal{X}$ as $f_{Bayes}(x) = argmax_y P(Y = y/X = x)$ and supervised learning attempts to build predictor $\hat{f}$ close to it based on samples $\{x_i, y_i\} \in \mathcal{X} \times \mathcal{Y}$.We denote $\mathcal{T} = \{\mathcal{Y}, P(Y/X)\}$ or $\mathcal{T} = \left\{\mathcal{Y}, \hat{f}\right\}$.*

To illustrate it in our example, simulated fall and real fall databases represent data coming from two domains on the same feature space $\mathcal{X}$ and the task to deal with is the fall detection on the same binary label space $\mathcal{Y} = \{0,1\}$ (corresponding to {*Non-fall*,*Fall*}). As it is observable that joint distribution $P(X,Y)$ over $(\mathcal{X},\mathcal{Y})$ varies while considering the two datasets it means that we are confronted to different yet related *domains* or different *tasks*. Usually the knowledge transfer is specifically directed from one domain to the other, then these are respectively referred as *Source* and *Target* domains.

Definition 3.3 (Transfer learning). *Denoting a source domain $\mathcal{D}^S = \{\mathcal{X}^S, P^S(X^S)\}$, a source task $\mathcal{T}^S = \{\mathcal{Y}^S, P^S(Y/X)\}$, a target domain $\mathcal{D}^T = \{\mathcal{X}^T, P^T(X^T)\}$ and a target task $\mathcal{T}^T = \{\mathcal{Y}^T, P^T(Y/X)\}$. Several domain differences are possible : $\mathcal{D}^S \neq \mathcal{D}^T$ (meaning $\mathcal{X}^S \neq \mathcal{X}^T$ or just $P^S(X) \neq P^T(X)$) or $\mathcal{T}^S \neq \mathcal{T}^T$ (meaning $\mathcal{Y}^S \neq \mathcal{Y}^T$ or just $P^S(Y/X) \neq P^T(Y/X)$). Transfer learning then attempts to use the knowledge from $\mathcal{D}^S$, $\mathcal{T}^S$ and $\mathcal{D}^T$ to improve the task $\mathcal{T}^T$.*

Definition 3.4 (Source/Target expected and empirical error). *Using back notations defined in Section 2 of Chapter 2, transfer learning context implies that, for a given predictor h, Source and Target expected error $R_{P^S}(h)$ and $R_{P^T}(h)$ differ, as well as for the empirical ones.*

Recent works study links between R_{P^S},R_{P^T} and empirical Source/Target errors in order to extend PAC-Bayesian theory to the transfer learning framework [96–98, 277]. Theoretical works on domain adaptation provided major answers to generic transfer learning questions such as : the way to quantify relevantly the dissimilarity between domains [166, 178], to insure the efficiency of knowledge transfer between domains under particular assumptions [20, 21, 190], or on the contrary to point out that some assumptions are not sufficient in certain situations [22]. Overviews of these theoretical results can be found in [149, 206] recent works.

There are several ways to classify main transfer learning frameworks. [191] defines frameworks based on the availability of labeled data in the source or target domain. Hence, when labeled data is available in the target domain, it is referred to as *inductive transfer*. Otherwise (i.e. when no labels are available in target), it is called *transductive transfer*. *Inductive transfer* is usually more focused on differences between tasks whereas *transductive transfer* is more about domain differences so it is also referred as *domain adaptation* which sometimes considers few labeled target samples (instead of no label at all). [257] use another categorization based on the equality (or not) between feature spaces, namely *homogeneous transfer learning* when $\mathcal{X}^S = \mathcal{X}^T$,$\mathcal{Y}^S = \mathcal{Y}^T$ and *heterogeneous transfer learning* otherwise. Heterogeneous transfer situations are also widely present in industrial applications with their own approaches [67] and could also be considered for Tarkett monitoring system as it aims at dealing with various tasks trained with

feature subsets that can differ, but it is out of this work's scope as it was not one of the emergency issues to solve.

These types of issues are very present in industrial application of machine learning and several of them can be enumerated based on the Tarkett system example as presented in Figure 3.3.

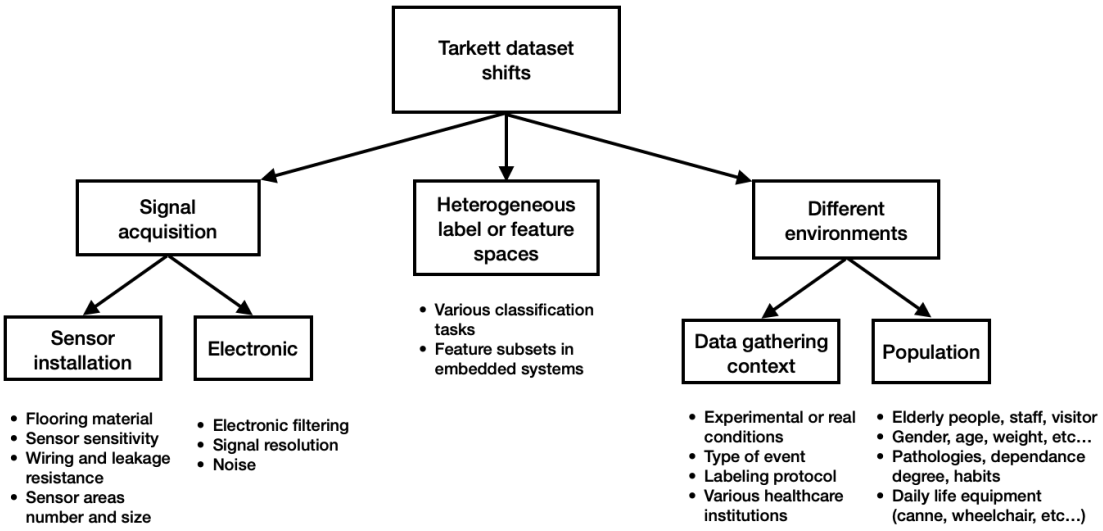


Figure 3.3: Possible use cases of transfer learning for Tarkett monitoring application. In this work we focus on the heterogeneity coming from the data gathering context. In particular, the transfer from simulated falls DB to real falls DB.

2.1 Transfer learning for human activity monitoring

Transfer learning field raised rapidly and concurrently to the increase of machine learning application using data from real environments. Indeed dealing with different but related domains or/and tasks is usual in various application such as images or videos classification [89, 103] or natural language processing [27, 49, 201]. Recently important an amount of works on transfer learning applied to human activity recognition is also noticeable. Comparably to the different domains and tasks for Tarkett monitoring system represented in Figure 3.3, the HAR application field regroups a lot of predictive tasks (as wide as the range of human activities to detect), plenty of different sensors (wearable, cameras, microphones, Wifi, smart-floors, etc...) installed in various kinds of environments. Sources of data dissimilarity are multiple, for instance [204] proposes a transfer method based on domain matching with data coming from different smart environments whereas [153] compares domain adaptation on radar spectrogram data with a domain shift due to people ages and sensor locations.

Considering the particular case of wearable sensors for HAR [50] proposes a comparison of different unsupervised domain adaptation methods in various situations. As some domain variation can be simulated by anticipation [225] presents a comparison of artificial data augmentation methods versus domain adaptation for activity recognition. [60] provides an extended survey on transfer learning for activity recognition but several works have been proposed since then, especially in the context of heterogeneous transfer [200, 214] and scalability with regards to data noise and sensor updates [9, 138].

2.2 Transfer learning methods and algorithms

Previous categories of transfer learning correspond to the big picture of the transfer learning problem contexts in terms of label information and equality or not of feature and label spaces, they represent what kind of data is available in source and target domains. Another finer categorization refers to the relations between source and target distributions, in other words what kind of assumption is done on these distributions to develop the transfer learning strategy.

Indeed if we consider homogeneous transfer learning where $(\mathcal{X}^S, \mathcal{Y}^S) = (\mathcal{X}^T, \mathcal{Y}^T)$, source and target difference relies on joint distribution $P^S(X,Y) \neq P^T(X,Y)$ and the nature of this difference can come from marginal or conditional distributions. While $P^S(X) \neq P^T(X)$ and $P^S(Y/X) = P^T(Y/X)$ it is referred as *covariate shift* and inverting the role of X and Y variables on this assumption is referred as *target shift* (while $P^S(Y) \neq P^T(Y)$ and $P^S(X/Y) = P^T(X/Y)$). On the contrary some works assume $P^S(X) = P^T(X)$ (resp. $P^S(Y) = P^T(Y)$) and $P^S(Y/X) \neq P^T(Y/X)$ (resp. $P^S(X/Y) \neq P^T(X/Y)$), which are referred as *conditional shift* [206] or *concept shift* [277]. Another assumption that can be reasonably done in several application is the *sample bias selection* when source domain distribution is submitted to a latent sample selection binary variable tending to exclude some data. The formulation of the joint distribution is then $P^S(X,Y) = P^S(q = 1/X,Y)P^S(X,Y)$ (with q the selection variable). It can be noticed that while q distribution depends only on X (resp. Y) this situation is equivalent to *covariate shift* (resp. *target shift*).

Finally a last and more practical categorization, that can also be found in the different evoked surveys, is more about possible approaches to handle transfer learning problems. They are tightly linked to the kind of assumptions on the source/target relation and are often regrouped in three main kinds of methods : *instance-based*, *feature-based* or *model-based* (a.k.a "parameter-based") methods.

Instance-based domain adaptation

Instance-based techniques basically rely on samples weighting methods through distributions estimation and are often linked with *covariate shift* assumption. Indeed it is by pointing out for a given predictor h the Target expected error can be expressed as :

$$R_{P^T}(h) = \mathbb{E}_{(x,y)\sim P^S}\left(w(x)\frac{P^T(y/x)}{P^S(y/x)}l(h(x),y)\right) \tag{3.1}$$

with $w(x) = \frac{P^T(x)}{P^S(x)}$, that [221] formulates the assumption of *covariate shift* under which $P^S(Y/X) = P^T(Y/X)$, meaning that expected target error can be simplified and viewed as a weighted version of the expected error on the source domain. Thus under this assumption plenty of instance weighting methods have been developed varying in the way of estimating the distribution ratio $w(x)$ [26, 105, 122, 227]. Intuitively the idea behind these approaches is to favor source instances that are close to target ones by high weights and decrease weights of source instances that are less similar to target. From that point numerous variants have been designed, for example [63] proposes an extension of the original AdaBoost algorithm [91] based on iterative estimations of instance weights depending on base learners predictions. [62] proposes an innovative instance weighting method independently to covariate shift assumption but dependent of the considered classifier, based on mean discrepancy minimization.

As previously precised instance based methods for covariate shift can also be applied for *sample bias selection* while the bias depends only on X variable. Moreover, by inverting symmetrically the role of variables X and Y in Equation 3.1, in *target shift* situations if target labels are available (which is rarely the case in domain adaptation), the same kind of approach can be used by estimating the other marginal distributions ratio : $w(y) = \frac{P^T(y)}{P^S(y)}$.

Feature based domain adaptation :

Feature-based transfer methods are approaches that apply operations on source or/and target feature spaces in order to find new feature representation where source and target dataset are more similar. It often supposes that there is a subset of feature shared between source and target domains or that there exists a mapping between them. For example in the context of natural language processing [27] suggests to first look for a latent space composed by the most frequent features (words in this application) present in both domains and named *pivot* features. Then these binary classifiers based on these pivots are trained to predict correlations between pivot features and other ones, and outputs of these predictors are considered as new features. This augmentation method based on shared features between source and target is named *structural correspondence* and has been also used for cross-lingual adaptation [201]. Another example is proposed in [66] perform feature augmentation where authors put up a feature space of dimension three times bigger than the original where the original vectors are stacked with 0-value vectors placed differently depending if the instance comes from source or target.

Other methods use what is named *feature alignment*, consisting in extraction subspaces from source and target using principal component analysis and projection of these subspaces on an intermediate one [89]. This kind of approach has the advantage to be applicable on heterogeneous feature spaces [9, 214] and to create a representation relevant for both source and target data through the intermediate subspace ensuring learning efficiency on both domains. In certain situations it can be reasonable to assume that target domain corresponds to a specific kind of transformation of source domain. For example [112] proposes translation and rotation based transformations to match datasets of multiple domains (not only two for source and target), and [251] proposes to match directly source and target distributions using kernel mean matching and linear transformations.

More recently, some works extended this source/target data mapping principle proposing a parallel to optimal transport theory. The main idea is to formulate domains mapping as a label transport problem over feature spaces between source and target distributions [90, 205] and to optimize it.

Model based transfer :

The last type of approach tackles the model trained on source to adapt it over target data. This set of methods is also known as *parameter-based* or *model-based* transfer learning. These methods usually make use both of target labeled data (inductive transfer) and a model previously trained on source data. An example of parameter-based algorithm is the adaptive-SVM [269] which uses target data to modify a SVM model already trained on source data. For this purpose, authors add a regularization

function to the previously trained decision function and minimize a trade-off between proximity with the original model and the predictive error along target data. Generally model-based transfer learning intends to keep source domain information through the structure or the parameters of an already trained model while inducing in it target domain information through transformations allowing it to fit better target data. From this perspective it is comparable to some approaches that deal with data shift in a data stream context. Indeed in this situation it is not affordable to build a completely new model but the original one can be adapted, in particular for ensemble classifier where base learners can be re-weighted [110], added or deleted [141] from the ensemble based on new data.

One big advantage of these methods is that they require only target data, this is why model-based transfer learning is particularly useful when source data is not available and this may be the case with privacy issues or data storage limitations. Concerning fall detection transfer learning from simulated data to real data, theoretic assumptions on relation between domain distributions can hardly be done a priori as causes of domains difference are likely to be multiple, complex and entangled. Moreover as presented previously in Figure 3.3, Tarkett monitoring system entails several different domains with several transfer learning situations. For these reasons evoked instance-based methods are excluded for our purposes as they usually rely on strong assumptions and are not very polyvalent to various transfer situations and unknown distribution changes. As previously explained, all the detection tasks for activity monitoring are implemented in a single embedded system for each patient. There is a first layer of signal processing, then a second one for features computations and a third one for random forest models. This last layer can be easily updated remotely to adapt the models whereas signal processing and feature computation parts are shared by all the detection models, are computationally optimized and hardly encoded in the system entailing that any change on these part has to be done "by hands" on each electronic hardwares. Thus using a feature-based transfer learning method is inconceivable in practice considering the design of our system.

For these reasons this work focuses on model-based methods on decision trees, that can be used with few labeled target data and without any need of source data, scalable to different transfer situations and that can ideally give interpretation clues about source/target data differences.

2.3 Transfer learning on decision tree based models

Several works on adaptation of decision tree based models to data shift in the context of data streams have been previously evoked in section 2.1 of Chapter 2. Even if they follow the same data shift adaptation principle, these decision tree based data stream methods are not dedicated to the transfer learning of a model from a source to a target domain as defined earlier but rather to keep up with continuous changes in data or successive occasional shifts, hence deviating from the transfer learning hypothesis (the target domain is not supposed to change and the target dataset is a fixed size batch of data).

An example of a proper model based transfer algorithm on decision trees is proposed in [151]. It considers the situation where target feature and label spaces can differ and updates a previously trained tree (on source data) by expanding nodes with new

features and re-affecting leaves label values while it is relevant, thus building a new model adapted to target domain and task from the previous source induced model. In the context of *covariate shift*, [235] proposes a random forest training algorithm using both source and target data weighted by an iterative covariate loss estimation that is comparable to the idea of transfer using instance weighting on an ensemble boosting method [160]. [103] investigates the situation of transferring one classification task from other related ones to increase overall performance and applies it on gesture recognition with random forest. To achieve this goal a variant of classical RF induction is proposed using two task-related notions : the mixed information gain for split nodes and the label propagation for leaves.

So this chapter focuses on two model-based transfer procedures on random forests detailed in the next section, with only a few labeled target data and a source model, with the hypothesis of same feature and label spaces. Our transfer learning context is then model-based inductive and homogeneous transfer on random forest classifiers.

3 Decision tree expansion, reduction and threshold updates for transfer learning

A model-based approach for model-based transfer using a RF trained on a source domain to a target domain based on the same feature space and predictive task has been investigated recently by [216] in a general and distribution assumptions free manner. Authors proposed two algorithms named STRUT and SER, that aim at adapting each tree from the original random forest, to improve the performance with respect to the target data. STRUT adapts the decision thresholds in order to cope with shifts, while SER modifies the tree structure in order to create more fine-grained or coarse-grained representations. In this section we describe the main operations these algorithms rely on and illustrate which kind of relation between source and target data they intend to cope with.

Framework and notations:

Given a source domain $\mathcal{D}^S$ and a target domain $\mathcal{D}^T$, source data are supposed to be unknown but we assume a source decision tree trained over data drawn from$\mathcal{D}^S$ is accessible, as well as few labeled target samples S^T assumed to i.i.d samples drawn from $\mathcal{D}^T$. Given a node v in the source decision tree, S^T_v denotes the subset of S^T that reaches v.

3.1 Structure Expansion/Reduction (SER)

SER is a recursive algorithm that applies two transformations relatively to target labeled data, in two successive steps : *Expansion* and *Reduction*.

First the *expansion* step consists of expanding any leaf (terminal node) l into a subtree if this node is not pure relatively to S^T_l samples (meaning that there are at least two labels reaching this leaf) . This subtree is computed by growing a new binary decision tree using CART algorithm from S^T_l samples.

Then the second transformation, the *reduction* aims at removing source tree unnecessary nodes, relatively to target samples, by pruning. It relies on the computation

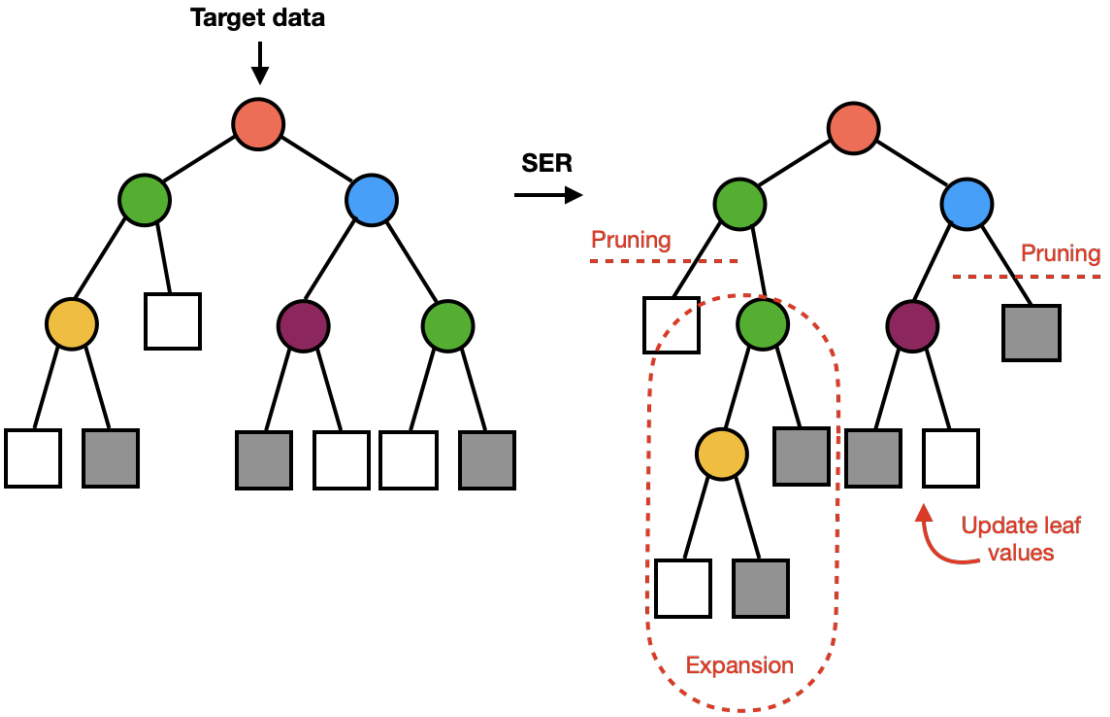


Figure 3.4: Illustration of *SER* operations on decision trees.

of the *leaf error* which, for any node v and it corresponding set S_v^T, is defined as the misclassification error at the current node:

$$LeafError(v, S_v) = \frac{1}{|S_v^T|} \sum_{k=1}^{|S_v^T|} \mathbb{1}_{\{y^{(k)} \neq y_v\}} ,$$

with $y^{(k)}$ the label of the k-th sample of S_v and y_v the majority class of node v. It cuts any node that has a leaf error lower to the *subtree error*. The subtree error relative to a subset S_v is the sum of all leaves errors weighted by proportion of S_v that reaches each leaf. In other words it is the error of the tree whose root is the node v:

$$TreeError(v, S_v) = \sum_{l \in L(v)} \frac{|S_l|}{|S_v|} LeafError(l, S_l) ,$$

where $L(v)$ is the set of leaves of the subtree that starts at node v.

Pruning : These errors are not explicitly given in the original paper but rather described as the empirical errors of the subtree (whose root is v) and the leaf (considering v as a leaf), hence the use of the classical misclassification error. Therefore, the *tree error* is in fact always zero, and we consider this step to be a pruning step that conserves the tree coherence for target data. Indeed, if a node v is not reached by target data, then the *leaf error* of v is considered to be zero and the pruning condition is respected. This is according to us the only case where pruning occurs, and it leads to the same condition developed in STRUT algorithm (see next section), which is motivated by the willing to obtain a tree that fits new data by removing unreachable parts of the tree.

The pseudo-code of SER is presented in Algorithm 3.

Algorithm 3.1 SER

procedure SER(v, S_v)
  % *Expansion*
  if v is a leaf **then**
    $v \leftarrow$ BuildTree(S_v)
    return v
  end if
  % *Recursive calls*
  SER($v_r, S_{v,r}$)
  SER($v_l, S_{v,l}$)
  % *Reduction*
  if LeafError(v, S_v) $\leq$ TreeError(v, S_v) **then**
    $v \leftarrow$ Prune(v)
  end if
  return v
end procedure

3.2 STRUT divergence optimization

The Structure Transfer (STRUT) algorithm goes through the decision tree from the top to the bottom applying the following procedure at each node v. If v is unreachable it is pruned into a leaf, otherwise, there are two possible situations : either it is a leaf and its output is then updated, or it is a regular node and the algorithm recomputes its threshold according to the set S_v^T that falls into it. Unlike a classical split procedure, here the measure that is optimized is different and there is no choosing in the feature (the algorithm updates the threshold while keeping the same variable).

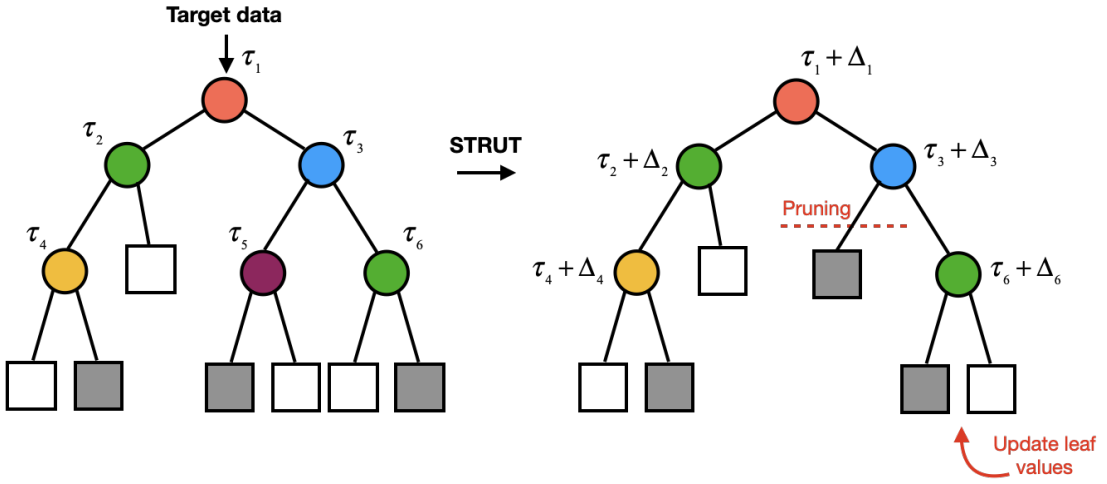


Figure 3.5: Illustration of *STRUT* operations on decision trees.
The different colors stand for different features and Δ_i updates to the node thresholds τ_i correspond to the divergence gain optimization.

The threshold selection procedure is done as follows. Given a node v, Q_l^S and Q_r^S denote respectively the class proportions of source data in left child node and right child

node after the original split. $Q_l^T(\tau)$ and $Q_r^T(\tau)$ denote respectively the class proportions of target data in left child node and right child node after the new split τ. As authors aim at finding a new threshold while keeping similar label distributions between source and target, they define a *divergence gain* (DG) that measures the similarity between the original label distributions (Q_l^S, Q_r^S) and the new ones (Q_l^T, Q_r^T). DG is defined as:

$$DG(S_v^T, \tau, Q_l^S, Q_r^S) = 1 - \frac{|S_{v,l}^T|}{|S_v^T|} JSD(Q_l^S, Q_l^T) - \frac{|S_{v,r}^T|}{|S_v^T|} JSD(Q_r^S, Q_r^T) \,. \tag{3.2}$$

$S_{v,r}^T$ and $S_{v,l}^T$ are the subsets of S_v^T that falls respectively into the right and left child node of v, and JSD is the Jensen-Shannon divergence, defined as:

$$JSD(P, Q) = \frac{1}{2} \left(KL(P, M) + KL(Q, M) \right) \,. \tag{3.3}$$

The Jensen-Shannon divergence measures the dissimilarity between the two distributions P and Q using the Kullback-Leibler divergence, denoted KL, and the mean distribution $M = \frac{1}{2}(P + Q)$. Hence, for a node v, DG measures the dissimilarity between previously learned source distribution and the newly set target distribution. We note that in practice, Q_l^S, Q_r^S, Q_l^T and Q_r^T are vectors that contain the proportions of each class at a given node. Hence we can write them $Q_l^S = \left(Q_{l,1}^S \dots Q_{l,K}^S \right)$, K being the number of classes.

The threshold selection procedure relies on an optimization problem that can be summarized as follows: the goal is to maximize DG while insuring a local maximum of the *information gain* (IG), defined here as the Gini gain. At node v, we denote $\Phi_v = \{\phi_1, \dots, \phi_N\}$ the set of all ordered distinct feature values of instances of S_v^T. Then, $\mathrm{T}_v = \{\tau_1, \dots, \tau_{N-1}\} = \{\frac{\phi_1+\phi_2}{2}, \dots, \frac{\phi_{N-1}+\phi_N}{2}\}$ represents the set of all possible thresholds considered by the *threshold selection*.

The selected τ denoted τ_m is then defined as:

$$\tau_m = \underset{\tau \in \mathrm{T}_v}{\arg\max}\, DG(S_v^T, \tau, Q_l^S, Q_r^S) \quad \text{s.t.} \quad \begin{cases} IG(\tau_{m-1}) < IG(\tau_m) \\ IG(\tau_m) > IG(\tau_{m+1}) \end{cases} \tag{3.4}$$

As pointed out by authors, between source and target, different labels can also swap relatively to the node threshold. To take into account this possible event during the *threshold selection*, the optimization problem has to be solved a second time swapping left and right for Q^S and Q^T which gives another threshold $\overline{\tau_m}$. Finally, the threshold selection procedure ends by taking the threshold maximizing DG between the two optimal thresholds : τ_m and $\overline{\tau_m}$. The pseudo-code of STRUT is given in Algorithm 3.2.

Algorithm 3.2 STRUT

procedure STRUT(v, S_v)
  if $|S_v| = 0$ **then**
    % *Prune unreachable node*
    $v \leftarrow$ Prune(v)
    return v
  else if v is a leaf **then**
    $v \leftarrow$ UpdateLeafValue(S_v)
    return v
  else
    % *Recompute threshold*
    $v \leftarrow$ ThresholdSelection(S_v, Q_l^S, Q_r^S)
    STRUT($v_r, S_{v,r}$)
    STRUT($v_l, S_{v,l}$)
  end if
end procedure

To sum up, SER seems to be well adapted when decision tree partitioning needs to be refined (expansion step) or on the contrary to be more coarse (reduction step). On the other hand, STRUT is especially designed for local axis drifts. Together, both procedures are meant to capture the possible transformations a decision tree may need to be transferred along target data. For this purpose original work [216] suggest to apply SER and STRUT to all trees of a random forest and then to mix both kinds of transferred trees in a new random forest.

To understand what SER/STRUT particularities imply several remarks can be done:

1. Both algorithms try to keep one aspect of source decision trees structure. SER keeps all nodes unchanged until either a node is not reached by any target data or it performs an extension at one original leaf. STRUT first keeps the feature unchanged at any node attempts to stay close to local class proportion induced by node split, through *divergence gain* threshold optimization.

2. SER and STRUT don't use the same amount of information about source distribution. Indeed at a given node v STRUT needs Q^S estimations of $p^S(Y/x \in S_v)$ whereas SER does not, meaning that this information has to be embedded in source model.

3. The possible decision function changes induced by the transfer seems wider with SER. For example STRUT can not add new node so it is limited in terms of depth and number of nodes. Moreover technically any new decision function can be fitted by SER as extensions are sufficient to create any new decision rule.

3.3 SER/STRUT depending on the type of transformation

In order to figure out which kind of source/target transformations are best handled by each of these two transfer algorithms we used a synthetic data generator to better control some simple data shifts. It is a basic Gaussian clusters distribution generator

and a Python implementation with its description is available online [4]. It will be used in this chapter and the next one for experimental purposes.

Three simple data shifts scenarios between source and target data have been implemented : a cluster mean drift situation, an addition of new clusters and a deletion of clusters. Behaviors of *SER* and *STRUT* algorithms on a unique decision tree classifier are presented in Figure 3.6. As expected *STRUT* outperforms *SER*, as well as source and target models, on cluster drift situation whereas *SER* is better to handle cluster addition and deletion. Nevertheless from these simple cases *SER* appears to be a bit more flexible as, even in drift situation, its performance difference with *STRUT* is not very significant.

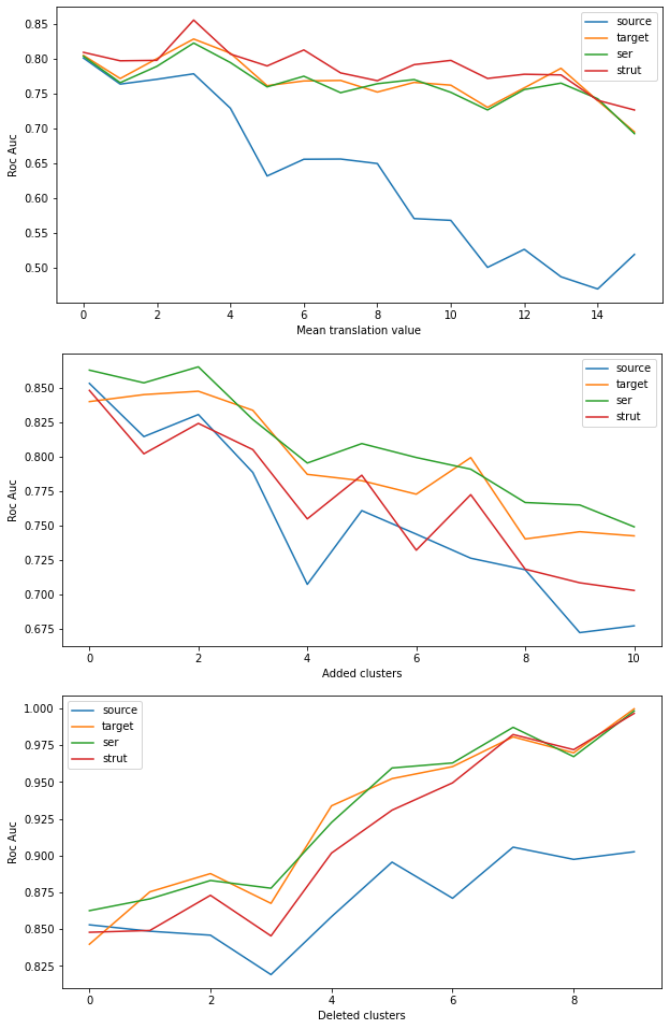


Figure 3.6: ROC AUC of *SER*/*STRUT* on unique decision trees with synthetic data on various conditions.
From top to the bottom : translation of clusters, new clusters addition and clusters deletion.

Of course in real application transformations between source and target are often not as simple as these synthetic examples. Moreover for transfer learning on fall detection application, we do not have any prior information about it except that one DB is balanced and the other is very imbalanced. In the next section we investigates

how class imbalance impacts transfer leaning, especially concerning *SER/STRUT* algorithms used on our fall detection data.

4 Class imbalance and pruning risk

Considering simulated fall DB as generated from a source domain and real fall DB from a target domain, as previously evoked the first observable difference even before computing any feature is the respective class proportions. Indeed fall detection problem is in real condition a highly imbalanced classification task and the source model has to be transferred taking into account this target class imbalance. In this section we explain why class imbalance increases classification task and transfer learning difficulty, as well as what drawbacks might be expected while using SER/STRUT with imbalanced target data on two core operations of these algorithms : *pruning* and *divergence gain* optimization.

Class	**Src**	**Targ**	**Ser**	**Strut**
Maj class	10.9	6.3	13.8	4.8
Min class	10.7	5.0	7.0	2.2

Table 3.1: Mean number of leaves for each class with different algorithms tested on synthetic data

4.1 Machine learning with class imbalance

Apart from fall detection numerous real ground classification applications are confronted to the imbalance problem especially anomaly detection ones and, in general, the imbalance can allude to two main situations [117]. First, class marginal distributions may be unequal, i.e. one class is largely under-represented compared to others. This may be due to a difficulty in the acquisition of a certain class in the data set (e.g. fraud detection, or rare event in time series). Secondly, error costs may be different depending on the class, this is the case when we aim at recognizing (or not) a class that has an arbitrary high importance (e.g. for medical diagnosis). Label dependent error costs occur very often in practice and several *cost-sensitive* adaptation techniques of machine learning exist [85], but they rely on a *error costs matrix* to be defined, which requires external knowledge about how critical is each type of error. Those two situations may also happen at the same time, i.e. we may have a rare event in the training data set and at the same time it may be crucial to detect it. Regardless the kind of imbalance situation this context is even harder to deal with in practice as it also implies performance assessment measure issues [18].

Considering class imbalance between marginal distributions, main methods to overcome this bias consist in sampling techniques. Most popular sampling methods use training data to select them in a way the class imbalance does not appear to the learning model. The most straightforward approaches are oversampling and undersampling [117] consisting in artificial minority class weight augmentation by repetitions and preponderant classes weight decrease ignoring some data, in order to compensate initial class imbalance. For instance [144] uses a procedure named "one-sided selection"

to under-sample majority class using random selection and nearest neighbors. The principle is to select only the hardest samples to classify among all majority class samples, but it is sensible to the random sample selection initialization and it does not ensure balanced dataset after this procedure. Sampling methods can also utilize data augmentation through generation of synthetic instances [52]. This kind of methods have yet some limitations such as discarding potential useful data, high variance or greater learning time.

So class imbalance is a machine learning problem on its own, but it can also interact with transfer learning while class imbalance is part of domain dissimilarities [10, 156], increasing learning difficulty. Indeed according to [258, 259], transfer learning under imbalanced domain is a challenging task that may lead to negative transfer. Authors define *domain class imbalance*, which differs from classical class imbalance in the fact that there is a variation in class probability between source and target data. It gathers situations where source might balanced and target imbalanced or reversely. As presented earlier if conditional probability $P_{X|Y}$ is unchanged between domains it is a particular case of *target shift* situation, which is already a challenging tasks when target labels are supposed to be unknown [279], as it is often the case in domain adaptation. In most practical cases and partly due to a training sample selection bias, one may have access to a relatively balanced data set for training a model that will perform on real data that is potentially imbalanced. Class imbalance also often comes along with relative rarity of the minority class instances, which is common in a transfer context where some target domain data may be hard to collect, as for fall detection.

This part focuses on the case where source data are balanced whereas target data are not. Moreover to cope with our fall detection situation we assume that only a few labeled target data are available, at least for the minority class. Indeed data rarity is tightly linked to the imbalance problem, but it also adds new practical data mining drawbacks [119, 256]. So here we study the impact of both class imbalance and data rarity on the two previously presented transfer algorithms SER and STRUT, particularly with regards to the risk of losing precious source domain information about the minority class.

4.2 Transfer with homogeneous class imbalance

Considering class imbalance, we study the specific effect of class proportion changes during transfer while source data are balanced whereas target data are highly imbalanced. This change could be due to *sampling selection bias* for example. Indeed in our case simulated DB follows an arbitrary distribution of different events and protocols hypothetically made to be representative of the majority of real events. On the contrary the real DB is composed by real condition events partly randomly picked for non-falls and a few falls acquired on the early version of the system and labeled manually and approximately. So this *sampling selection bias* is obvious in our situation and is likely to be at least one part of the explanation concerning class proportion change. It can be defined with a binary random variable ζ representing sampling selection as follows :

$$p^S(X,Y) = p(\zeta^T = 1/X,Y)p(X,Y)$$
$$p^T(X,Y) = p(\zeta^S = 1/X,Y)p(X,Y)$$

Here source and target distributions p^S and p^T come from the same underlying probability p and their difference relies on different sampling strategies between source and target.

Notations. $\mathcal{D} = (\mathcal{X}, \mathcal{Y}, p)$ denotes a domain, with $\mathcal{X}$ and $\mathcal{Y}$ respectively its feature and label spaces, and a probability distribution p over $(\mathcal{X}, \mathcal{Y})$. In transfer learning, two domains are considered: a source domain $\mathcal{D}^S = (\mathcal{X}^S, \mathcal{Y}^S, p^S)$ and a target domain $\mathcal{D}^T = (\mathcal{X}^T, \mathcal{Y}^T, p^T)$. We consider in this paper $\mathcal{X}^S = \mathcal{X}^T$ and $\mathcal{Y}^S = \mathcal{Y}^T$. ${P_y}^S = \int p^S(x,y)dx$ and ${P_y}^T = \int p^T(x,y)dx$ respectively denote the source and target marginal distributions over $\mathcal{Y}$.

Definition 3.5 (Homogeneous class imbalance)**.** *Given a source domain $\mathcal{D}^S = (\mathcal{X}^S, \mathcal{Y}^S, p^S)$ and a target domain $\mathcal{D}^T = (\mathcal{X}^T, \mathcal{Y}^T, p^T)$, homogeneous class imbalance occurs when:*

- *Source domain is balanced :* $\forall y_1, y_2 \in \mathcal{Y}, P^S_{y_1} \simeq P^S_{y_2}$
- *Target domain is imbalanced :* $\exists y_1, y_2 \in \mathcal{Y}, P^S_{y_1} \ll P^S_{y_2}$
- *The imbalance change is homogeneous over the feature space:*

$$\forall x, y \in (\mathcal{X}^S \cap \mathcal{X}^T, \mathcal{Y}^S \cap \mathcal{Y}^T), p^T(x/y) = p^S(x/y) \tag{3.5}$$

Firstly equation 3.5 equivalently means (using Bayes' rule) :

$$p^T(y/x) = w(y)\frac{p^S(y/x)}{\sum_{y'} w(y')p^S(y'/x)} \tag{3.6}$$

with $w(y) = \frac{P^T_y}{P^S_y}$ being the ratio between target and source label proportion for each label y.

Moreover equation 3.6 is exactly the symmetric assumption of *covariate shift* with regards to variables x, y. So it implies that, under this assumption and following the same idea as in equation 3.1 but inverting x ans y variables, the *target expected* error can be expressed as :

$$R_{P^T}(h) = \mathbb{E}_{(x,y)\sim P^S}\left(w(y)l(h(x), y)\right) \tag{3.7}$$

for a given classifier h and loss function l.

In domain adaptation this assumption is commonly known as *target shift*. Nevertheless contrary to the usual domain adaptation scheme we have access to target labels, meaning that $w(y)$ quantities can be directly estimated on target data.

It should be precised that here, the term *homogeneous* is not related to *homogeneous transfer learning* but rather means that the class proportion change is homogeneous throughout the feature space since it depends only on $w(y)$. It can be seen as the "simplest" target class imbalance situation and will be used as a reference to study what kind of impacts can be expected while performing transfer learning under class imbalance. From this point of view any transfer algorithm has to be able to handle at least *homogeneous class imbalance* to be considered suited for imbalanced target domains.

4.3 Data rarity and the pruning risk

As previously explained both SER and STRUT algorithms use pruning to deal with situations where some nodes are unreached by any target data. In this context we formulate the pruning risk as the probability to lose a relevant minority class leaf because of this pruning, i.e. the risk for a minority class leaf to be pruned even if it is still representative for the target domain.

Definition 3.6. *Let k_{min} denote the minority class and L a leaf of class k_{min} from a source decision tree. The leaf L is still significant or relevant for the target domain if:*

$$\forall k \neq k_{min}, \quad p^T(y = k_{min}/x \in L) > p^T(y = k/x \in L) \tag{3.8}$$

Definition 3.7. *Pruning risk Let n_k be the amount of class k target data available and assume that these data are drawn i.i.d from the same p^T distribution. We define the pruning risk relative to the minority class leaf L as the probability of L being unreached by any of the n_k minority class data, which is then :*

$$PRR_{n_{k_{min}}}(l) = p^T(x \notin L/y = k_{min})^{n_{k_{min}}} \tag{3.9}$$

In balanced conditions and out of data rarity situations, $n_{k_{min}}$ is large enough so the probability tends to approach 0 and the pruning risk becomes negligible. However when $n_{k_{min}}$ decreases, it leads to higher quantity of pruned leaves. Let us now consider the case where the only transformation that occurs between source and target is an *homogeneous class imbalance.* Under this assumption and simply using (3.5), (3.8) and (3.9) can be reformulated as follows:

$$\forall k \neq k_{min}, \lambda_{k_{min}} p^S(y = k_{min}/x \in l) > \lambda_k p^S(y = k/x \in l) \tag{3.10}$$

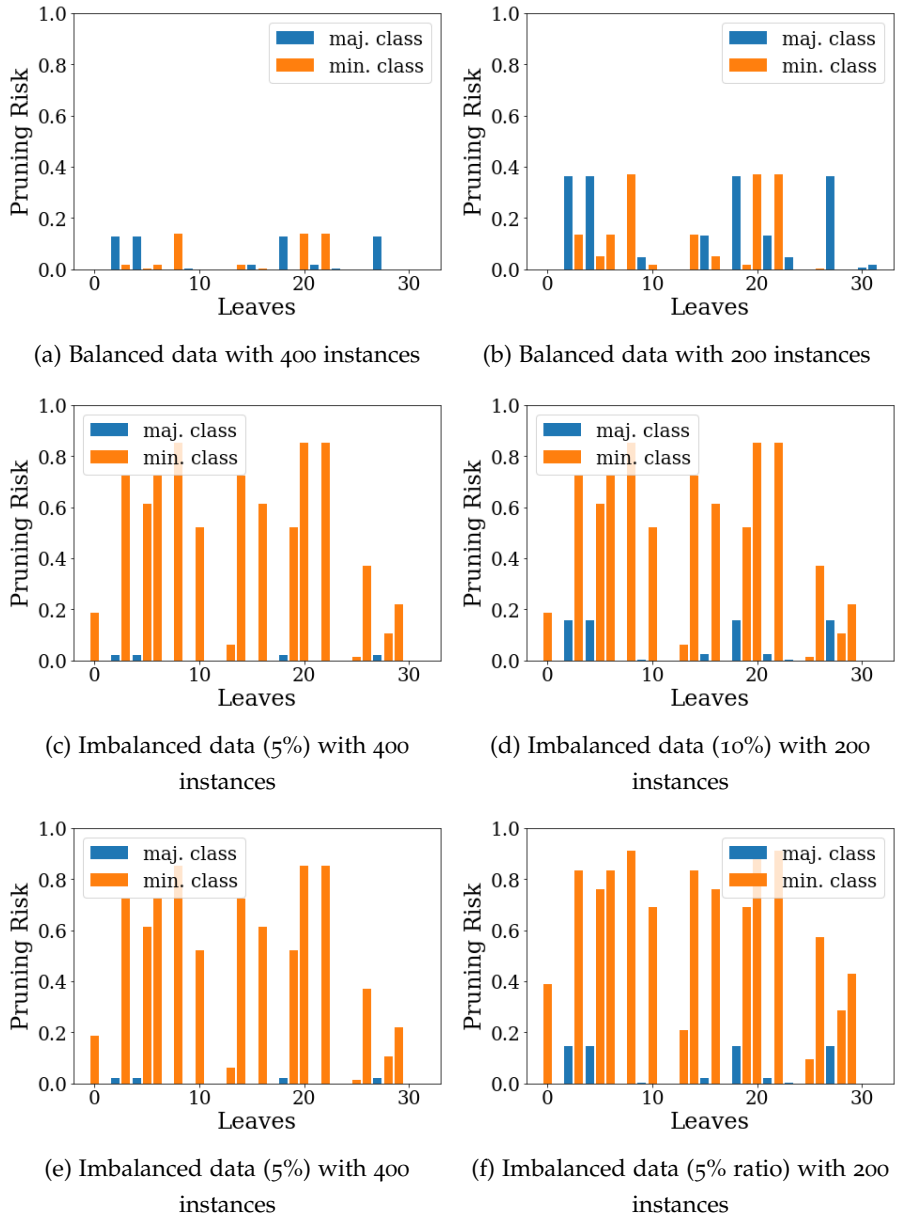
$$PRR_{n_{k_{min}}}(l) = p^S(x \notin l/y = k_{min})^{n_{k_{min}}} \tag{3.11}$$

Although this result relies on a theoretical assumption that is not necessarily verified in practice, computing this value can give some insights about the impact of using pruning algorithms such as SER and STRUT on minority leaves. Figure **??** shows example of leaf loss risk computation using (3.11) under various conditions.

This means that while class imbalance entails minority class rarity for training the transfer algorithm, pruning operation may result in discarding relevant source leaves, even in the basic situation of homogeneous class imbalance. It is not negligible as both SER and STRUT use pruning on regions that are unreached by target data. In section 5, we present our strategies to limit this pruning risk based on its estimation and on structural consistency conservation.

4.4 Divergence gain optimization under class imbalance

The divergence gain (DG) defined for the STRUT procedure measures the similarity between previous class distributions (Q_l^S, Q_r^S) and the new ones (Q_l^T, Q_r^T). As previously said, STRUT algorithm recomputes thresholds from the top to the bottom of the tree using DG (and IG). Hence at a given node, when the threshold is recomputed, the label distributions according to source data (Q_l^S and Q_r^S) change. However, as we consider



(a) Balanced data with 400 instances

(b) Balanced data with 200 instances

(c) Imbalanced data (5%) with 400 instances

(d) Imbalanced data (10%) with 200 instances

(e) Imbalanced data (5%) with 400 instances

(f) Imbalanced data (5% ratio) with 200 instances

Figure 3.7: Pruning risk for a decision tree under different target data conditions, with two classes.

Blue ones are **majority class** leaves and orange ones correspond to **minority class**. With balanced target data (a), the risk of losing minority leaves is similar to the risk of losing majority class leaves. By decreasing the number of target data (b), the risk increases equally regardless of the leaf class. However when dealing with imbalanced data (c), the leaf loss risk on minority leaves is significantly higher, and in the same imbalance conditions while decreasing the number of target data (d), the risk is even worse.

that we do not have access to the source data during the transfer procedure, it means that we cannot recompute Q_l^S and Q_r^S according to the new threshold.

As we go deeper into the tree while performing STRUT, Q_l^S and Q_r^S that correspond to previous threshold values are more likely to mislead the algorithm, especially if distributions were swapped (left and right) as described in Section 3.2. Indeed, at

a given node v, computing Q_l^S and Q_r^S implies to know where the source data falls according to all thresholds that lead to this node (i.e. the thresholds of all parents nodes of node v).

Besides, trying to stay as close as possible to the source node classes distributions implies the assumption that at each node, class distributions have not much evolved between source and target. However in our case this is the opposite: we have to transfer a source model built on balanced data with a target distribution that is imbalanced. To show the influence of DG in the performances, a version of STRUT *without using DG* is proposed and tested along with all other algorithms.

5 Model based transfer on random forest with adaptation to class imbalance

These two issues, i.e. the pruning risk and the divergence gain optimization issue in $STRUT$, lead us to adapt the initial SER and $STRUT$ algorithms to target data class imbalance. In this section we present several variants of these algorithms based on *homogeneous imbalance* assumption and pruning avoidance strategy in order to limit class imbalance drawbacks. Python implementation of $SER/STRUT$ and the proposed variants : $SER_{NoPrune}$, $SER_{NoPrune}(\lambda)$, $STRUT_{NoDiv}$, $STRUT(\lambda)$ and $STRUT_{NoPrune}(\lambda)$ are available online [1].

5.1 SER adaptation : class-dependent pruning

Firstly we developed a variant $SER_{NoPrune}$ (or SER_{NP}) specifically for target class imbalance situation which limits drastically minority class pruning. As previously pointed out, the main issue concerning SER come from the pruning relevant minority class leaves when there are too few target data of a class, leading to unfavorable pruning. Actually, for a given source leaf L, during the transfer procedure, there are three possible events relatively to target training data:

1. L is reached by target training data and keep the same majority class as source
2. L is reached by target training data and the majority class changes (i.e. it differs from the source majority class)
3. L is not reached by any target training data

Case 1 is the most favorable event (i.e. the minority class leaf is conserved). In practice, considering SER algorithm, the two other cases correspond to the two transfer steps, i.e. *expansion* (case 2) and *reduction* (case 3). To impact on minority leaf loss events, SER_{NP} focuses only on the *reduction* step. While confronted to case 3 on a minority class leaf, it assesses whether this leaf would be still significant under *homogeneous imbalance* (using Equation 3.10) and if applicable then avoids the pruning of the source leaf. However it keeps the *expansion* phase on them allowing finer partitioning from a source minority class leaf.

This approach may seem harsh since it conserves leaves that were defined uniquely with source data and not any target sample. However if we consider that doing transfer with imbalanced data biases the model towards the majority class, this method aims to

compensate this bias towards the minority class. Pseudo-code of SER_{NP} is given by Algorithm 3.3.

Algorithm 3.3 $SER_{NoPrune}$

procedure $SER_{NP}(v, S_v)$
  % Expansion
  if v is a leaf **then**
    $v \leftarrow \text{BuildTree}(S_v)$
    return v
  end if
  % Recursive calls
  $SER_{NP}(v_r, S_{v_r})$
  $SER_{NP}(v_l, S_{v_l})$
  % Reduction
  if $\text{LErr}(v, S_v) \leq \text{STErr}(v, S_v)$ **then**
    if $c_m \in y(S_v^S)$ **then**
      % Don't prune
    else
      $v \leftarrow \text{Prune}(v)$
    end if
  end if
  return v
end procedure

Notations: c_m is the minority class

Although being drastic, $SER_{NoPrune}$ is really efficient to not lose too many minority leaves during the transfer learning (see Table 3.2) and further results show its performance on extreme class imbalance situations. Nevertheless this variant should be used only in this type of specific situation while source minority class leaves are somehow considered reliable even for target domain.

So in order to come up with a more general and flexible version of SER_{NP} another variant following the same pruning avoidance principle but where the confidence in source minority class leaves can be adjusted through a threshold parameter compared with the pruning risk. This algorithm, namely $SER_{NoPrune}(\lambda)$ ($SER_{NP}(\lambda)$), computes the pruning risk estimation PRR under homogeneous imbalance using Equation 3.11 and determines pruning avoidance situation if this estimated risk is higher than a threshold C_{PRR}. Thus $SER_{NoPrune}(\lambda)$ is equivalent to original SER algorithm while $C_{PRR} \simeq 1$ and is the same as SER_{NP} while $C_{PRR} \simeq 0$, providing a generalization of SER. Pseudo-code of SER_{NP} is given by Algorithm 3.4.

5.2 STRUT adaptation: divergence gain optimization with homogeneous imbalance

As explained previously, class ratio changes are not directly compatible with the use of the divergence DG in the $STRUT$ algorithm. For this reason, a version in which the optimization problem does not use DG is tested. In this configuration, it simply

Algorithm 3.4 $SER_{NoPrune}(\lambda)$

procedure $SER_{NP}(\lambda)(v, S_v, C_{PRR})$
  % Expansion
  if v is a leaf **then**
    $v \leftarrow \text{BuildTree}(S_v)$
    return v
  end if
  % Recursive calls
  $SER_{NP}(\lambda)(v_r, S_{v_r})$
  $SER_{NP}(\lambda)(v_l, S_{v_l})$
  % Reduction
  if $\text{LErr}(v, S_v) \leq \text{STErr}(v, S_v)$ **then**
    if $c_m \in y(S_v^S)$ **then**
      $T_v \leftarrow SubTree(v)$
      $\mathcal{L} \leftarrow \text{HomImbKminLeaf}(T_v, \lambda)$
      if $|\mathcal{L}| > 0$ and $\exists l \in \mathcal{L} : \text{PruningRisk}(n_{k_{min}}, \lambda) > C_{PRR}$ **then**
        % Don't prune
      else
        $v \leftarrow \text{Prune}(v)$
      end if
    end if
  end if
  return v
end procedure

Notations: c_m is the minority class

consists of updating thresholds by recomputing the maximum Gini gain over target data while keeping the same feature. It is referred to as $STRUT_{NoDiv}$ (or $STRUT_{ND}$).

Another concern is that our transfer methods need to be able to handle at the very least the homogeneous class imbalance transformations. Equation (3.6) in a discrete formulation becomes :

$$p^T(y = k/x) = \lambda_k \frac{p^S(y = k/x)}{\sum_i \lambda_i p^S(y = i/x)} \tag{3.12}$$

Our $STRUT$ adaptation method, replaces Q_l^S and Q_r^S using Equation 3.12 in the divergence optimization, thus considering that the homogeneous class imbalance condition is satisfied. For each class k, $Q_{l,k}^S$ and $Q_{r,k}^S$ become:

$$Q_{l,k}^{S\,*} = \frac{\lambda_k Q_{l,k}^S}{\sum_i \lambda_i Q_l^S} \qquad Q_{r,k}^{S\,*} = \frac{\lambda_k Q_{r,k}^S}{\sum_i \lambda_i Q_r^S} \tag{3.13}$$

We note that in this configuration, if we consider a transfer procedure in which classes proportions are conserved between source and target, then for each class k we have $\lambda_k \simeq 1$. This leads to $Q_{l,k}^{S\,*} \simeq Q_{l,k}^S$ which goes back to the original STRUT algorithm,

Algorithm 3.5 $STRUT(\lambda)$

procedure STRUT$(\lambda)(v, S_v)$
  if $v = root$ **then**
    $n_{k_{min}} = |\{(x,y) \in S_v / y = k_{min}\}|$
    % Compute λ
    for each class k :
      $\lambda_k = \frac{p_k(S^T)}{p_k(S^S)}$
  end if
  if $|S_v| = 0$ **then**
    $v \leftarrow$ Prune(v)
    return v
  end if
  if v is a leaf **then**
    $v \leftarrow$ ChangeLeafValue(S_v)
  else
    $(Q^{Imb}_{v_l}, Q^{Imb}_{v_r}) \leftarrow$ TargetShift$(Q^S_{v_l}, Q^S_{v_r})$
    $v \leftarrow$ ThSel$(\phi_v, S_v, {Q_{v_r}}^I, {Q_{v_r}}^I)$
    STRUT$(\lambda)(v_r, S_{v_r})$
    STRUT$(\lambda)(v_l, S_{v_l})$
  end if
  return v
end procedure

hence showing that it is a generalization of STRUT. This algorithm is named $STRUT(\lambda)$ and its pseudo-code is given by Algorithm 3.5.

5.3 STRUT pruning limitation and path consistency

The previous $STRUT(\lambda)$ algorithm is meant to limit the impact of class marginal probability changes on the DG optimization problem. Actually any important change in class proportion between source and target domain can induce some drawbacks on DG optimization and *target shift* assumption can help to compensate them. Class imbalance in target is a particular case of this situation and *homogeneous class imbalance* a particularization of *target shift*. But as explained previously another issue linked to class imbalance is relative data rarity and the associated pruning risk.

$STRUT(\lambda)$ has a local impact at each node on DG optimization but is not very efficient to limit pruning although having a slight effect as shown in Table 3.2. Thus here is presented another $STRUT$ variant that actively counters pruning risk. As well as for $SER_{NP}(\lambda)$ minority leaves with a high pruning risk can be tagged in order to avoid their pruning but there is a major difference while delaing with $STRUT$ algorithm. Indeed, contrary to SER, $STRUT$ uses two structural operations that can provoke nodes path consistency issues : split threshold changes and sub-tree swaps. The original $STRUT$ algorithm does not suffer from this problem as each new node threshold is recomputed using reaching data after having modified its previous path, and if a swap creates unreached regions in the tree they are simply pruned. However preventing

pruning by trying to keep some nodes unchanged can break this path consistency.

So being able to avoid pruning in some parts of a tree require to solve this tree structure issue : how to modify a node threshold without any target data while this node is not consistent anymore with the path leading to it due to previous threshold updates. As no data are available to estimate any metric like Gini gain or DG in this situation, because source data are assumed unavailable and STRUT prunes a node only while no target data are reaching it, we can only consider a geometrical criteria.

Let's consider an unreached node v splitting a feature ϕ_v with an initial source threshold τ_v^S. In the initial source tree v is associated with a subspace of $\mathcal{X}$ with a projection along ϕ_v axis of the form : 1) $X_{\phi_v} =]-\infty, \tau_M^S]$, 2) $X_{\phi_v} = [\tau_m^S, +\infty[$ or 3) $X_{\phi_v}^S = [\tau_m^S, \tau_M^S]$. With τ_m^S, τ_M^S coming from previous splits on the path of v and these thresholds may have been updated into new ones τ_m^T, τ_M^T by STRUT procedure changing $X_{\phi_v}^S$ interval into a new one $X_{\phi_v}^T$.

Then path consistency issue occurs while $\tau_v^S \notin X_{\phi_v}^T$ and the geometrical criteria we implemented to solve it relies on the closest bound distance conservation for infinite size intervals and the relative threshold position conservation for finite intervals. In other words τ_v^T is obtained according to the form of $X_{\phi_v}^S$, following these rules :

1) $\tau_M^T - \tau_v^T = \tau_M^S - \tau_v^S$

2) $\tau_v^T - \tau_m^T = \tau_v^S - \tau_m^S$

3) $\frac{\tau_v^T - \tau_m^T}{\tau_M^T - \tau_m^T} = \frac{\tau_v^S - \tau_m^S}{\tau_M^S - \tau_m^S}$

This allows to design a conditional pruning version of STRUT based on the *pruning risk* estimation as for $SER_{NoPrune}(\lambda)$ but this time using these path consistency modifications, namely $STRUT_{NoPrune}(\lambda)$ (or $STRUT_{NP}(\lambda)$). Moreover while avoiding in this way to prune a node v, this consistency issue can concern several subnodes from this node v, this threshold translation criteria is then applied recursively on all the sub-tree starting from v. This operation is referred as Translate(T_v) in the following pseudo-code 3.6 presentation of $STRUT_{NoPrune}(\lambda)$ algorithm.

5.4 Experiments

5.4.1 Synthetic data

In order to easily control data transformations between source and target, we generate several data sets with three dimensional binary labeled clusters in a bounded space. Moreover, each synthetic source or target data set consists of a mixture of several weighted Gaussian clusters, each one corresponding to a single class. Source data consists of $N_{source} = 200$ samples, with balanced class distribution (i.e. 50% of each class) and drawn out of $N_{clust} = 15$ equally weighted Gaussian clusters for each label. The initial mean μ and variance σ^2 parameters of these Gaussian distributions are evenly randomly drawn between fixed bounds ($\mu \in [-50, 50]$, $\sigma^2 \in [5, 15]$).

From a given source data set, we construct the associated target data by applying homogeneous class imbalance (ratio going from 3% to 25%), combined with transformations of tree kinds: 1) Drifts: change in Gaussian clusters means, 2) Squeezes/Stretches:

Algorithm 3.6 $STRUT_{NoPrune}(\lambda)$

procedure $STRUT_{NP}(\lambda)(v, S_v, C_{PRR})$
  if $v = root$ **then**
    $n_{k_{min}} = |\{(x,y) \in S_v / y = k_{min}\}|$
    % Compute λ
    for each class k :
      $\lambda_k = \frac{p_k(S^T)}{p_k(S^S)}$
  end if
  if $|S_v| = 0$ **then**
    $T_v \leftarrow SubTree(v)$
    $\mathcal{L} \leftarrow$ HomImbKminLeaf(T_v, λ)
    if $|\mathcal{L}| > 0$ and $\exists l \in \mathcal{L}$: PruningRisk$(n_{k_{min}}, \lambda) > C_{PRR}$ **then**
      $v \leftarrow$ Translate$(\mathrm{T_v})$
      return v
    else
      $v \leftarrow$ Prune(v)
      return v
    end if
  end if
  if v is a leaf **then**
    $v \leftarrow$ ChangeLeafValue(S_v)
  else
    $(Q^{Imb}_{v_l}, Q^{Imb}_{v_r}) \leftarrow$ TargetShift$(Q^S_{v_l}, Q^S_{v_r})$
    $v \leftarrow$ ThSel$(\phi_v, S_v, {Q_{v_r}}^I, {Q_{v_r}}^I)$
    $STRUT_{NP}(\lambda)(v_r, S_{v_r}, C_{PRR})$
    $STRUT_{NP}(\lambda)(v_l, S_{v_l}, C_{PRR})$
  end if
  return v
end procedure

change in Gaussian clusters variances, 3) Redrawing randomly some clusters parameters. Then a new dataset of 200 samples is generated according to new parameters and used for the training of transfer algorithms. One advantage of these synthetic Gaussian controlled scenarios is that it is possible to generate as much data as needed for the evaluation process, which ensures good performance measure precision. Thus scores are assessed after generating 5000 new target data and each experiment is repeated 10 times. Versions using pruning risk estimation ($SER_{NP}(\lambda)$ and $STRUT_{NP}(\lambda)$) to avoid some minority class leaves pruning, the C_{PRR} threshold is set to 0.5.

Class	**Src**	**Targ**	SER	SER_{NP}	$SER_{NP}(\lambda)$	STR	STR_{ND}	$STR(\lambda)$	$STR_{NP}(\lambda)$
Maj class	10.9	6.3	13.8	15.5	13.5	4.8	4.5	7.9	6.9
Min class	10.7	5.0	7.0	9.8	7.7	2.2	2.3	3.4	5.8

Table 3.2: Mean number of leaves of each class using $SER/STRUT$ variants on synthetic data.

These experiments on synthetic and real data compare the original SER/STRUT algorithms and their proposed adaptations SER_{NP}, $SER_{NP}(\lambda)$, $STRUT(\lambda)$ and $STRUT_{NP}(\lambda)$ under class imbalance and rarity of the minority class in target data set. While dealing with these two conditions, choosing a performance measure that does not penalize the minority class, unlike accuracy for example, is a known issue [18]. As ROC curves consider all the trade-offs between false positive rate and true positive rate, they are commonly computed to extract performance assessment metrics in class imbalance situations [117, 256]. For instance, [10, 238] use area under ROC curves as a reference metric in the context of transfer learning with imbalanced conditions.

In all experiments, models trained only with source data or trained with only target data are also tested. Indeed, when performing transfer learning, one should make sure the transferred model outperforms the original one (trained on source data) and the model trained with the few available target data. In results those models are respectively referred to as *Source* and *Target*.

Transf.	**Src**	**Targ**	SER	SER_{NP}	$SER_{NP}(\lambda)$	**STRF**	**C**
I (5%)	0.874 ± 0.04	0.822 ± 0.02	0.883 ± 0.02	0.940 ± 0.02	0.912 ± 0.02	0.930 ± 0.02	$\frac{1}{3}$
I (10%)	0.861 ± 0.04	0.861 ± 0.02	0.910 ± 0.02	0.942 ± 0.02	0.928 ± 0.02	0.927 ± 0.02	
μ (5%)	0.684 ± 0.06	0.817 ± 0.02	0.828 ± 0.02	0.900 ± 0.02	0.879 ± 0.02	0.849 ± 0.02	
μ (10%)	0.635 ± 0.06	0.875 ± 0.02	0.924 ± 0.02	0.942 ± 0.02	0.927 ± 0.02	0.915 ± 0.02	
σ (5%)	0.841 ± 0.04	0.791 ± 0.02	0.839 ± 0.01	0.894 ± 0.01	0.879 ± 0.01	0.881 ± 0.01	$\frac{1}{3}$
σ (10%)	0.840 ± 0.04	0.871 ± 0.02	0.896 ± 0.01	0.923 ± 0.01	0.918 ± 0.01	0.919 ± 0.01	
μ,σ (5%)	0.658 ± 0.10	0.823 ± 0.02	0.859 ± 0.01	0.897 ± 0.01	0.883 ± 0.01	0.832 ± 0.02	
μ,σ (10%)	0.650 ± 0.10	0.827 ± 0.02	0.869 ± 0.01	0.893 ± 0.01	0.877 ± 0.01	0.855 ± 0.02	
I (5%)	0.676 ± 0.10	0.801 ± 0.01	0.859 ± 0.01	0.904 ± 0.01	0.881 ± 0.01	0.828 ± 0.02	$\frac{2}{3}$
I (10%)	0.670 ± 0.10	0.846 ± 0.01	0.891 ± 0.01	0.934 ± 0.01	0.918 ± 0.01	0.892 ± 0.02	
μ (5%)	0.535 ± 0.08	0.742 ± 0.02	0.761 ± 0.02	0.831 ± 0.02	0.821 ± 0.02	0.741 ± 0.02	
μ (10%)	0.573 ± 0.08	0.835 ± 0.02	0.857 ± 0.02	0.882 ± 0.02	0.876 ± 0.02	0.862 ± 0.02	
σ (5%)	0.741 ± 0.06	0.833 ± 0.02	0.853 ± 0.01	0.906 ± 0.01	0.887 ± 0.01	0.870 ± 0.02	$\frac{1}{3}$
σ (10%)	0.709 ± 0.06	0.878 ± 0.02	0.909 ± 0.01	0.924 ± 0.01	0.910 ± 0.01	0.899 ± 0.02	
μ,σ (5%)	0.564 ± 0.12	0.752 ± 0.02	0.785 ± 0.01	0.845 ± 0.01	0.844 ± 0.02	0.766 ± 0.01	
μ,σ (10%)	0.535 ± 0.12	0.867 ± 0.02	0.893 ± 0.01	0.913 ± 0.01	0.903 ± 0.02	0.875 ± 0.01	

Table 3.3: ROC AUC of transfer algorithms on various experiments with synthetic data. SER variants comparison.

Tables 3.3 and 3.4 represent transfer learning results on synthtic data in terms of ROC AUC over several repetitions and is focused on two imbalance ratios (5% and 10%). Each row corresponds to a specific transformation with a percentage of minority class. At each experiment either one third or two thirds of source clusters parameters have been redrawn to create target data. The last column recalls this portion. Each redrawing is combined with changes on mean (μ), variance (σ), or only class proportion changes (I). The best result within the STRUT and SER family methods are highlighted, unless differences with respect to the other variants are minor. Figure 3.8 gives the mean performance scores over all experiments for different proportions of minority class.

In most experiments, $STRUT$ does negative transfer which gets worse when more clusters are changed in the transformation. When regarding $STRUT_{NoDiv}$ mean score over all experiments, we observe that it always gives poorer results than Target, thus doing negative transfer. However when comparing it to $STRUT$, we observe that

Transf.	Src	Targ	$STRUT$	$STRUT_{ND}$	$STRUT(\lambda)$	$STRUT_{NP}(\lambda)$	STRF	C
I (5%)	0.874 ± 0.04	0.822 ± 0.02	0.751 ± 0.03	0.736 ± 0.04	0.776 ± 0.03	0.804 ± 0.03	0.930 ± 0.02	$\frac{1}{3}$
I (10%)	0.861 ± 0.04	0.861 ± 0.02	0.800 ± 0.03	0.786 ± 0.04	0.835 ± 0.03	0.828 ± 0.03	0.927 ± 0.02	
μ (5%)	0.684 ± 0.06	0.817 ± 0.02	0.732 ± 0.05	0.759 ± 0.04	0.800 ± 0.03	0.740 ± 0.03	0.849 ± 0.02	
μ (10%)	0.635 ± 0.06	0.875 ± 0.02	0.765 ± 0.05	0.737 ± 0.04	0.857 ± 0.03	0.780 ± 0.03	0.915 ± 0.02	
σ (5%)	0.841 ± 0.04	0.791 ± 0.02	0.694 ± 0.04	0.659 ± 0.04	0.750 ± 0.03	0.776 ± 0.03	0.881 ± 0.01	$\frac{1}{3}$
σ (10%)	0.840 ± 0.04	0.871 ± 0.02	0.745 ± 0.04	0.773 ± 0.04	0.814 ± 0.03	0.794 ± 0.03	0.919 ± 0.01	
μ,σ (5%)	0.658 ± 0.10	0.823 ± 0.02	0.681 ± 0.04	0.667 ± 0.04	0.780 ± 0.03	0.734 ± 0.03	0.832 ± 0.02	
μ,σ (10%)	0.650 ± 0.10	0.827 ± 0.02	0.725 ± 0.04	0.731 ± 0.04	0.804 ± 0.03	0.735 ± 0.03	0.855 ± 0.02	
I (5%)	0.676 ± 0.10	0.801 ± 0.01	0.691 ± 0.02	0.661 ± 0.03	0.750 ± 0.02	0.709 ± 0.02	0.828 ± 0.02	$\frac{2}{3}$
I (10%)	0.670 ± 0.10	0.846 ± 0.01	0.730 ± 0.02	0.724 ± 0.03	0.831 ± 0.02	0.734 ± 0.02	0.892 ± 0.02	
μ (5%)	0.535 ± 0.08	0.742 ± 0.02	0.639 ± 0.05	0.643 ± 0.04	0.676 ± 0.05	0.642 ± 0.05	0.741 ± 0.02	
μ (10%)	0.573 ± 0.08	0.835 ± 0.02	0.752 ± 0.05	0.759 ± 0.04	0.806 ± 0.05	0.704 ± 0.05	0.862 ± 0.02	
σ (5%)	0.741 ± 0.06	0.833 ± 0.02	0.731 ± 0.03	0.747 ± 0.05	0.797 ± 0.03	0.763 ± 0.03	0.870 ± 0.02	$\frac{2}{3}$
σ (10%)	0.709 ± 0.06	0.878 ± 0.02	0.800 ± 0.03	0.810 ± 0.05	0.853 ± 0.03	0.791 ± 0.03	0.899 ± 0.02	
μ,σ (5%)	0.564 ± 0.12	0.752 ± 0.02	0.662 ± 0.03	0.676 ± 0.04	0.692 ± 0.03	0.609 ± 0.03	0.766 ± 0.01	
μ,σ (10%)	0.535 ± 0.12	0.867 ± 0.02	0.748 ± 0.03	0.731 ± 0.04	0.820 ± 0.03	0.728 ± 0.03	0.875 ± 0.01	

Table 3.4: ROC AUC of transfer algorithms on various experiments with synthetic data. $STRUT$ variants comparison.

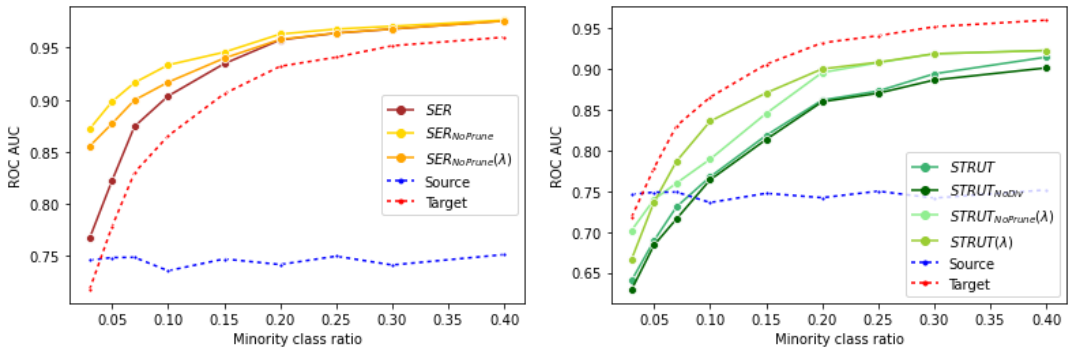


Figure 3.8: Mean ROC AUC of transfer methods on synthetic data depending on the minority class ratio.

its scores are always better, letting us assume that using the divergence criteria in the threshold update procedure worsen the transfer. We can observe that $STRUT(\lambda)$ scores stay relatively close to Target's independently to minority ratios. Nevertheless, experiments show that this algorithm still largely outperforms original $STRUT$ and $STRUT_{NoDiv}$.

Theses results suggest that when facing class imbalance, rather than ignoring the divergence criteria ($STRUT_{NoDiv}$) which is not effective, taking into account the class proportion change (with $STRUT(\lambda)$) is a better alternative. $STRUT_{NoPrune}(\lambda)$ transfer efficiency is comparable to $STRUT$ in most situations, except for extreme imbalance ($< 5\%$) where this variant represents a clear advantage.

Unlike $STRUT$, SER manages to do a valid transfer, i.e. it performs better than source and target models. Up to around 10% of minority class ratio, $SER_{NoPrune}$ outperforms significantly the original SER algorithm and $SER_{NoPrune}(\lambda)$ transfer efficiency is placed somewhere between the two, depending on the value of λ.

5.4.2 Transfer from simulated to real falls

Our source model is a 10 trees random forest trained on *simulated falls* data as presented in chapter 2. The *real falls* DB considered as our available target data is composed by 2465 event signals with 167 falls and 2298 non-fall events corresponding to random real activity signals. Transfer learning algorithms experiments have been made with ROC AUC as the performance metric and according to 2 main axis of observation : the *class imbalance ratio* and the absolute *amount of minority class data* used in transfer algorithms. Each transfer algorithm is applied on every tree of the source random forest producing a new transferred random forest of the same size. Moreover for versions that use pruning risk estimation ($SER_{NP}(\lambda)$ and $STRUT_{NP}(\lambda)$) to avoid some minority class leaves pruning, the C_{PRR} threshold is set to 0.5.

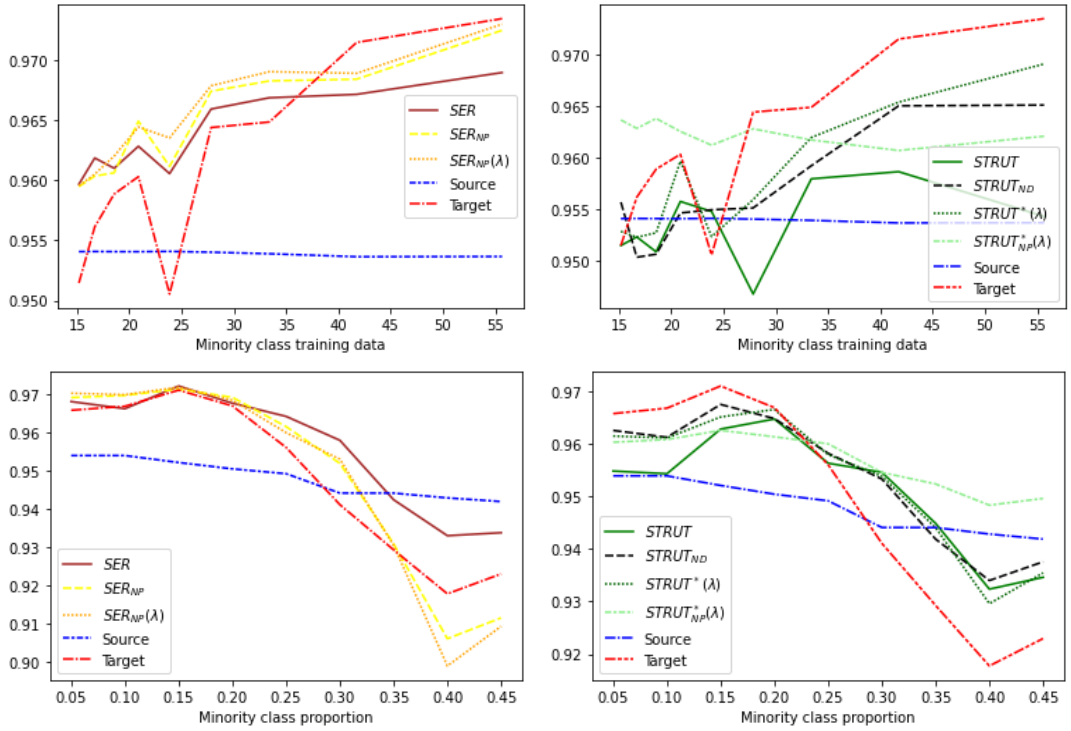


Figure 3.9: Mean ROC AUC of transfer algorithms on fall detection data depending on the minority class ratio and the amount of minority class data.

For each transfer experiment scores correspond to the mean ROC AUC of 10 repetitions of k-fold cross validation. To obtain a varying amount of minority class data the number of folds varies from 3 to 10 while keeping the same imbalance ratio ($\sim 7\%$), making the number of training target falls between 15 and 55. In order to get a varying class imbalance ratio while keeping the same amount of training falls non-fall events had to be added partially in the training such as falls represent from 3% to 45%. In this configuration the number of folds is 5 and the amount of training falls is around 30.

Results of these experiments are shown in Figure 3.9 in function of class imbalance ratio and number of training falls, grouped by algorithms type (SER or $STRUT$) and compared with source and target models. We first can observe that with the original 7% target class imbalance ratio and with a few falls event $STRUT_{NoDiv}$ version can

outperform $STRUT$, showing the limits of original $STRUT$'s *divergence gain* optimization in class imbalance situations. While varying the number of training falls target model is better than every $STRUT$ version from a certain point (around 30 falls) but our $STRUT(\lambda)$ and $STRUT_{NoPrune}(\lambda)$ variants provide some improvements compared to original $STRUT$. Moreover with less than 30 falls $STRUT_{NoPrune}(\lambda)$ outperforms quite distinctly all other models proving its efficiency to limit bad effects of pruning.

SER versions present in majority better scores than $STRUT$ ones, achieving a valid transfer with a performance noticeably higher than source and target models while confronted with very few minority class data. Pruning limitation of SER variants for imbalance slightly improve results and $SER_{NoPrune}(\lambda)$ performance curve seems a bit above $SER_{NoPrune}$ one, suggesting that pruning risk quantification can be more relevant than systematically avoid minority class pruning.

5.4.3 Public datasets

To confirm the validity of the variants we propose in the imbalance context three publicly available data sets of different nature have also been tested. One of them is composed by images and the two others are textual data.

Spam detection:
This data set contains 7500 text instances collected from messages and emails for spam detection purposes and provided originally for the *ECML/PKDD 2006 Discovery Challenge*. Two data sets are available online[1] (namely "Task A" and "Task B") and we used the one described as having important user specific characteristics to perform transfer learning ("Task A"). This database originally includes some unlabeled data but we extracted only labeled emails from three inboxes corresponding to three users (2500 instances for each user) referred as user "A","B" and "C".

For each experiment we consider one user as the *source* domain and another as the *target* domain and we generated features through word embedding keeping the amount of occurence of the 100 most frequent words. This situation is a generalization type of transfer between two different sources of the same kind of data. To make a parallel with our monitoring application it would be comparable to a transfer learning from a detection model trained on one patient room into another patient room.

Amazon reviews:
This data set is also composed by text data with binary labels from Amazon products reviews. Amazon provides millions of client online reviews on dozens of products and sorted databases on these data are available online[2]. Four categories of products are selected : music, video games, cell phones and home products and 5000 reviews (1250 for each kind of product) are extracted from this database for our experiments. These data are originally labeled by a "review score" going from 1 to 5 that we divided into two main labels : "positive" and "negative" reviews.

For each experiment we consider one product category as the *source* domain and another category as the *target* domain and we generated features through word embedding using the 500 most frequent words. This situation is a generalization type of

[1] www.ecmlpdd2006.org/challenge.html
[2] https://snap.stanford.edu/data/web-Amazon.html

transfer between two different groups of data sources. An analogy with our monitoring application would be to attempt a transfer between two groups of patient, sorted for example by sex, pathology, level of dependence or walking device.

Office-Caltech:

This data set contains images from amazon.com or office environment images taken under different conditions (with a webcam or a higher quality camera). Features are generated by SURF algorithm categorized in 800 dimensions available online[3] and images are partitioned in 10 classes corresponding to different objects. We use Amazon and Caltech image sets as *source* and Webcam images as *target*. The Amazon and Caltech sets contains respectively 958 and 1123 instances (about 100 examples per class) and the Webcam set contains 295 instances with (about 30 examples per class). To keep a simple binary label context we performed one versus all re-labeling for each experiment.

This is comparable to our transfer learning situation on fall detection task as it is an example of a *source* domain composed by data acquired in a controlled environment (here images for sales purposes with good resolution and uniform background) and a *target* domain composed by real environment data (here low quality images taken in different conditions of exposition etc.)

Results :

Figures 3.10, 3.11 and 3.12 provides respectively ROC AUC scores over spam detection data, Amazon reviews and Office-Caltech data for various class imbalance values in the target data set. For each transfer experiment scores correspond to the mean ROC AUC of 10 repetitions of k-fold cross validation on the target test set with k set to 5. The source class ratio is initially set to 50% and the imbalance ratio varies from 5% to 50% of minority class using downsampling, we set the imbalance ratio in the target set going from 5% to 50% of minority class. Figure 3.10 represents transfer experiment on spam detection task from user "A" to user "B", from user "B" to user "C" and from user "A" to user "C". Figure 3.11 represents transfer experiment on reviews classification task from digital music reviews to home equipment products reviews and from digital music to smartphones. Figure 3.12 represents transfer experiment on object recognition task from Amazon pictures to webcam ones and from Caltech to webcam, using one versus all mean score assessment. For algorithms using pruning risk estimation ($SER_{NP}(\lambda)$ and $STRUT_{NP}(\lambda)$) the C_{PRR} threshold is set to 0.5.

Concerning SER and its variants in most cases scores of these algorithms follow target models scores in function of class imbalance ratio but with noticeable better performance. It is not very surprising as the *expansion* phase is a sort of partial re-training with target data at some particular nodes of the decision trees. It explains why SER presents in general better results than $STRUT$ with the possibility of increasing depth of decision trees, contrary to $STRUT$, to cope with parts of target data distribution that would be totally new compared to source distribution.

Moreover SER and its variants have performance curves that are relatively close which suggests that the pruning applied by SER is not a big issue on these three transfer applications. On Office-Caltech data it is even preferable to chose the original

[3] https://people.eecs.berkeley.edu/~jhoffman/domainadapt/

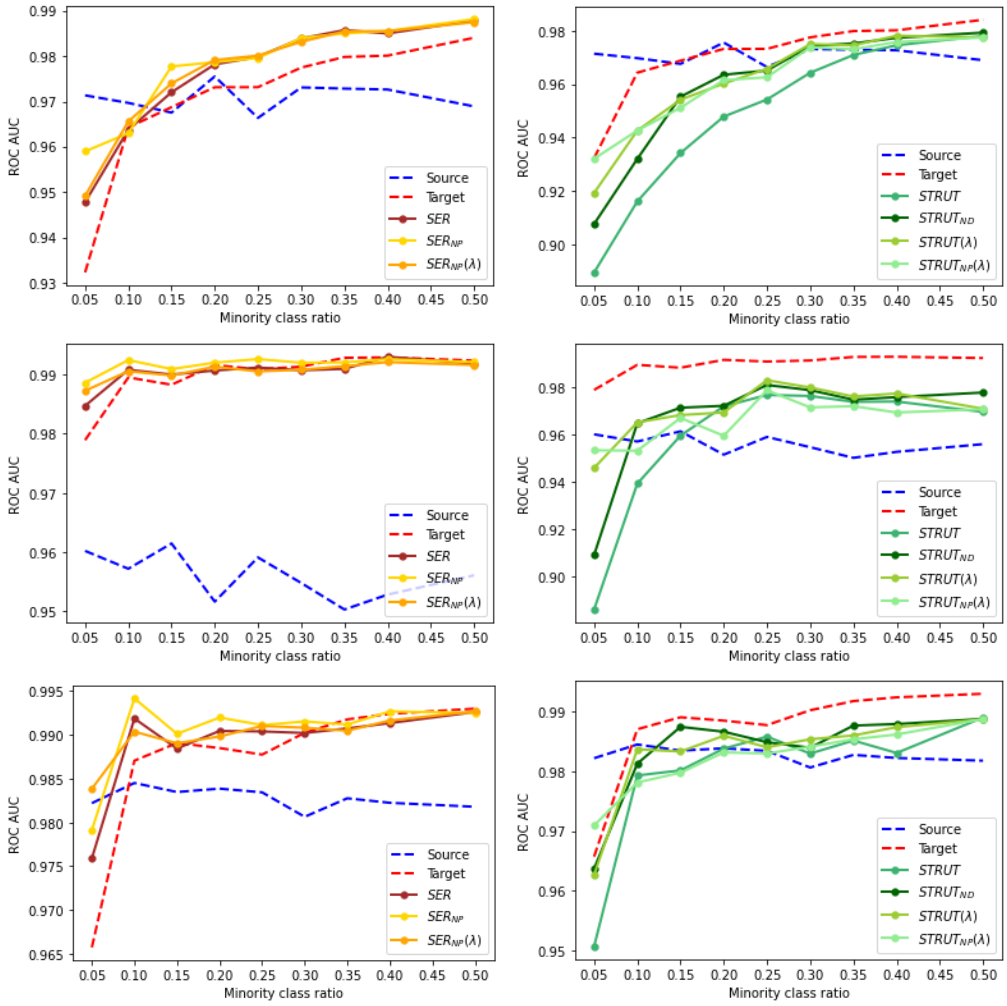


Figure 3.10: Mean ROC AUC of transfer methods on spam detection DB depending on the minority class ratio. First row : from user "A" to user "B". Second row : from user "B" to user "C". Third row : from user "A" to user "C".

SER algorithms (Figure 3.12). On the other hand, on the spam detection task presented in Figure 3.10, a slight improvement brought by SER_{NP} and $SER_{NP}(\lambda)$ is observable on low class imbalance ratios. Another interesting phenomenon on these data is that in some cases $SER_{NP}(\lambda)$ is better than SER_{NP} and the other in some other cases. As SER_{NP} corresponds to the extreme situation of $C_{PRR} = 0$, it means that the calibration of this pruning risk threshold parameter is important. Based on these observations we can conclude that *SER* is suited for transfer in a large variety of context and data and the effects of pruning avoidance variants with regards to class imbalance depends on the data set.

Observations on *STRUT* and its variants are less ambiguous, even if they present poorer performance than *SER* and than target models in most cases, the effect of the two main variants $STRUT(\lambda)$ and $STRUT_{NP}(\lambda)$ is clearly positive while confronted with important class imbalance.

Additional results on public data are presented in Appendix C.

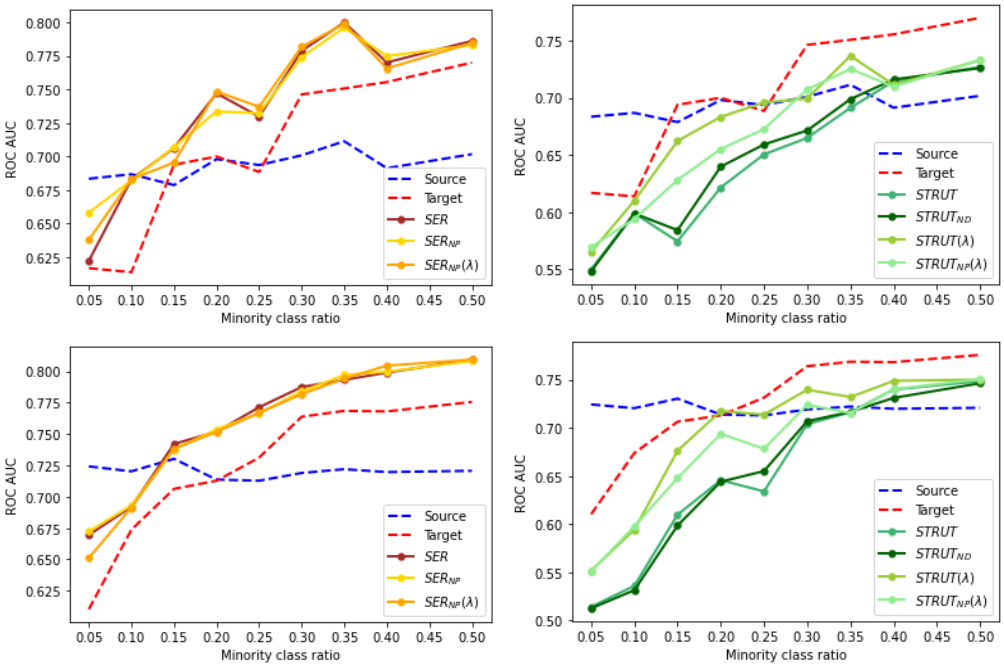


Figure 3.11: Mean ROC AUC of transfer methods on Amazon reviews DB depending on the minority class ratio.
First row : from *music* products to *home* products. Second row : from *music* products to *phone* products.

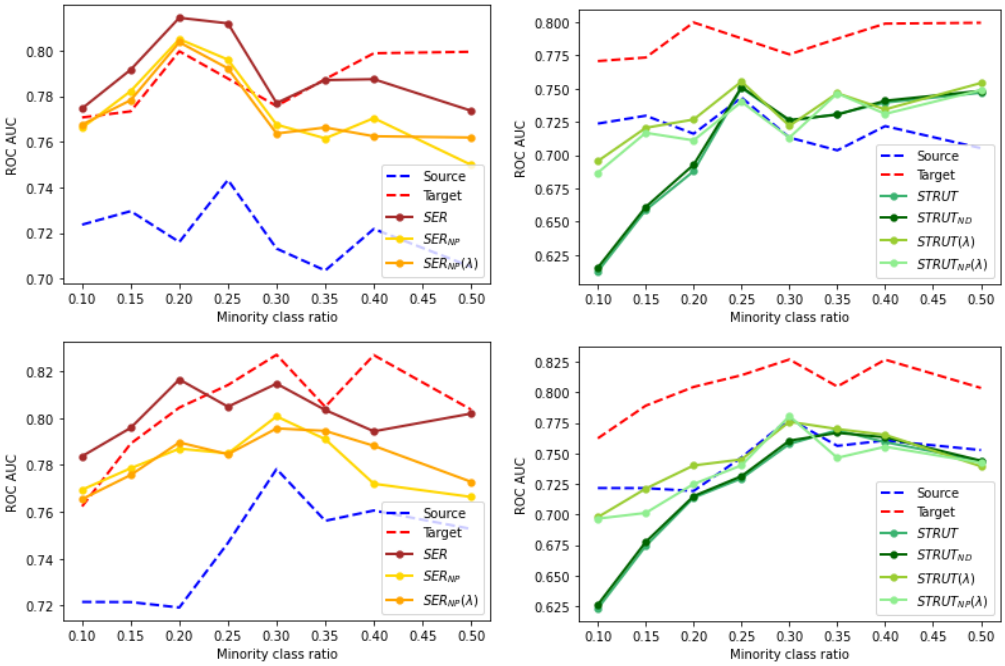


Figure 3.12: Mean ROC AUC of transfer methods on Office-Caltech object recognition DB depending on the minority class ratio.
First row : from *Amazon* images to *webcam* images. Second row : from *Caltech* images to *webcam*.

6 Meta-model and selection of transferred trees

In previous section we presented SER and $STRUT$ model-based transfer algorithms on decision trees and showed using synthetic and real datasets that their efficiency depends on data and the transformation between source and target domains. We explained the drawbacks of these algorithms while confronted to imbalanced target data, which is unavoidable while dealing with fall detection, and designed some variants adapted to class imbalance and reducing the pruning risk. Nevertheless, even if our variants can be seen as generalization of original $SER/STRUT$ algorithms, for any given set of target samples and decision tree, there is probably a best version of these algorithms to employ which is not known in advance. In their original work [216] authors suggest to apply both algorithms on every decision tree of the source random forest and to mix the resulting transferred trees in a new random forest two times bigger than the source one, containing then one half of SER decision trees and one half of $STRUT$ ones. They also showed that this combination is useful to increase performance as SER trees are non-correlated to $STRUT$ trees.

So we propose an aggregation method able to adapt to different transfer situations, taking into account all the various variants previously presented in a random forest of the same size of the original one, following two main ideas.

1. For each tree of the original random forest is selected a bootstrap subset of target data, keeping the primitive way of random forest to reduce correlation between trees.

2. Each transfer algorithm variant is trained using bootstrap samples and assessed using OOB samples and the best one is selected as the transferred version of the corresponding original tree.

6.1 Adaptable transfer algorithm for random forest

STRF algorithm : *Selective Transferred Random Forest* (STRF) algorithm is a meta-algorithm using several transfer sub-algorithms to deal with any transfer situation on random forests and to give information about source/target domains relation, its pseudo-code is presented in 3.7. Consider a source random forest, a labeled target dataset S^T and an ensemble of several transfer algorithms on decision trees $\mathcal{A} = \{A_1, ..., A_m\}$) designed to be efficient in different conditions. For example in our case we have an algorithm designed to adapt to distribution enrichment or simplification (SER) and another one designed to deal with local feature space geometric transformations (STRUT), as well as variants of these for imbalance and data rarity situations.

The algorithm works as follows : for each source tree T_i in the initial random forest a bootstrap subset S_b is drawn from S^T and used to train every transfer algorithms of $\mathcal{A}$, while OOB samples are kept to assess score of each one applied to T_i and to select the best transferred tree T_i^{tr}. We choose to use the ROC AUC score in this *SelectBest* procedure but it can be replaced by any score measure. Finally the transferred random forest corresponds to the aggregation of all selected transferred trees T_i^{tr}.

Algorithm 3.7 STRF

procedure STRF($RF, S^T, \mathcal{A} = \{A_1, ..., A_m\}$)
  for $T_i \in RF$ **do**
    $S_b, OOB_b \leftarrow$ Bootstrap(S^T)
    for $T_i \in RF$ **do**
      for $A_k \in \mathcal{A}$ **do**
        $T_i \leftarrow A_k(T, S_b)$
      end for
      $T_i^{tr} \leftarrow SelectBest(\{T_i^1, ..., T_i^m\}, OOB_b)$
    end for
  end for
  return $RF^{tr} \leftarrow \{T_i^{tr}\}_i$
end procedure

Notations: Here the list of different transfer algorithms on decision trees tested is $SER, SER_{NoPrune}, SER_{NoPrune}(\lambda), STRUT, STRUT(\lambda), STRUT_{NoDiv}, STRUT_{NoPrune}(\lambda)$. But other transfer algorithms can be used.

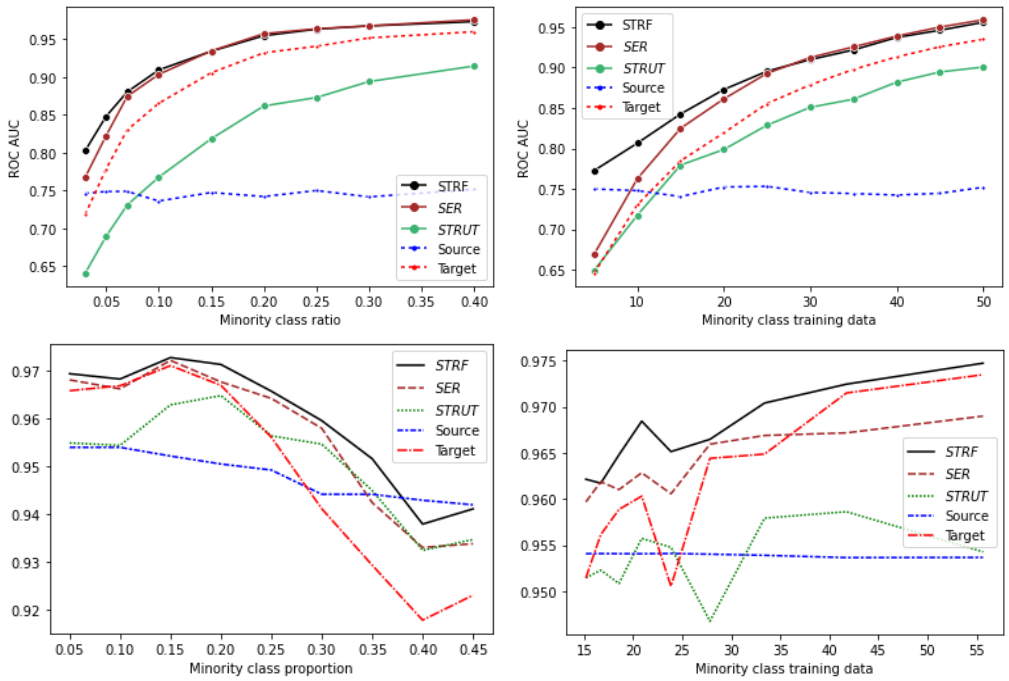


Figure 3.13: Comparison of STRF model with original $SER/STRUT$ on synthetic (top) and fall (bottom) data, depending on target minority class ratio and the amount of minority data.

This algorithm keeps original bagging idea of random forests and at the same time use transfer algorithms on decision trees in combination with OOB performance estimation to choose $SER/STRUT$ variants that are more suited to target data. This principle can be implemented on random forest with any group of transfer algorithms applicable to decision trees and should be the more efficient the less decision trees

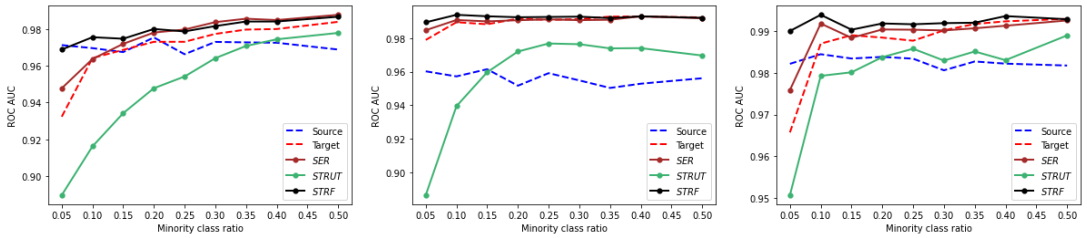


Figure 3.14: Comparison of STRF model with original $SER/STRUT$ on **spam detection** data, depending on target minority class ratio and the amount of minority data.
Transfer between inboxes of 3 user "A","B" and "C", respectively (A,B),(B,C) and (A,C) from left to right.

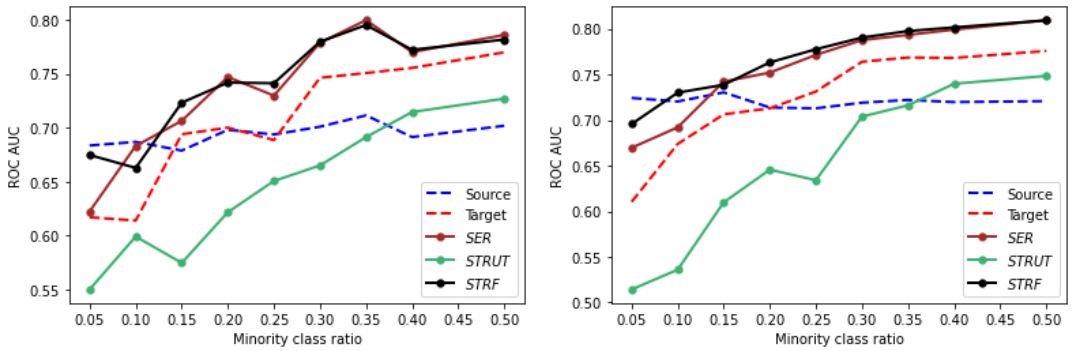


Figure 3.15: Comparison of STRF model with original $SER/STRUT$ on **Amazon reviews** data, depending on target minority class ratio and the amount of minority data.
Transfer from *music* to *home* domain (left) and from *music* to *phone* (right).

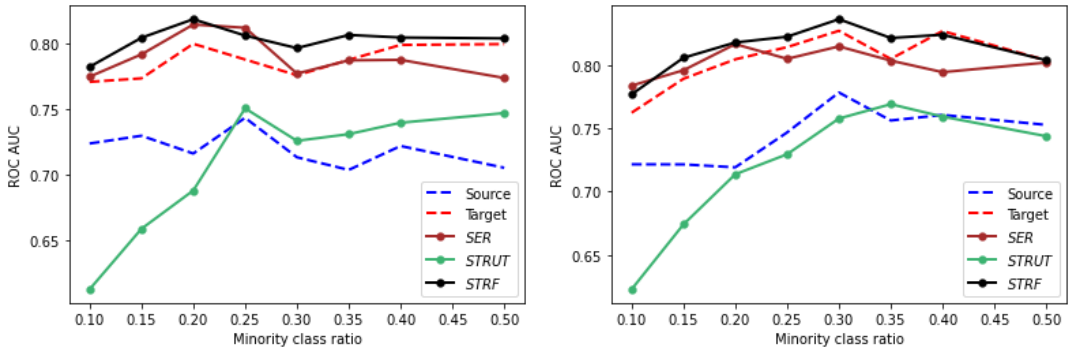


Figure 3.16: Comparison of STRF model with original $SER/STRUT$ on **Office-caltech** data, depending on target minority class ratio and the amount of minority data.
Transfer from *Amazon* (left) or *Caltech* images to *webcam* images(right). Average on one-vs-all models between 10 labels.

produced by the different transfer algorithms are correlated.

The main inconvenient of this STRF algorithm how heavy in terms of computations it can be as this procedure tests several transfer algorithms on each decision tree. Nevertheless this kind of transfer operation is meant to be done after new data gathering that shows major changes in data distributions and is not needed in real-time. So it is not a critical issue as long as the model complexity is comparable between the

transferred random forest and the source one.

6.2 Interpretation of STRF selection

Previous section shows the performance of STRF algorithm both on synthetic and real datasets. Most of the cases our aggregation transfer algorithm outperforms each individual decision tree transfer algorithm while staying relatively close to the best variant of $SER/STRUT$ depending on data, in terms of ROC AUC scores.

So in addition to providing good transfer performance STRF shows its ability to several kinds of source/target transformation, without any prior knowledge, by a sub-model selection strategy. Moreover as this algorithm can be considered as a kind of "vote" for the best transfer $SER/STRUT$ variant on decision trees, it indicates in a way which kind of transfer operations are preferable to choose depending on data, which can be relevant information. For example we can assume that if STRF chooses in majority $STRUT$ variants it could mean that the relation between source and target domains is made of local translations. On the contrary, if STRF votes mainly for SER variants it may mean that there are some new data clusters in target domain, as SER can perform tree extension whereas $STRUT$ can not augment trees depth. Synthetic experiments on unique decision trees show these kinds of tendencies as presented in Figure 3.6.

Extending this data interpretation idea we can wonder whether this selective transfer algorithm could detect situations at the borders of transfer learning like while target domain comes from a completely new distribution or *negative transfer* cases. Indeed the literature presents several prepared dataset where transfer is known to be possible and useful but in reality one may be confronted to the problem of transfer utility and necessity. To study these situations we used synthetic data to produce several experimental situations in combination with adding source and target (RF trained from scratch on target data only) random forests in the pool of models that can be selected by the STRF algorithm.

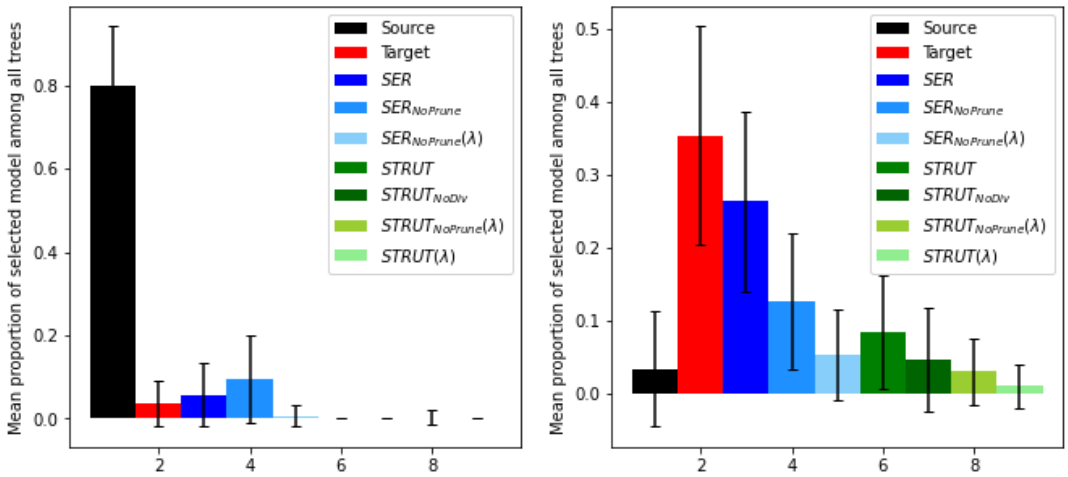


Figure 3.17: Visualization of STRF selection on synthetic data (situations 1. and 2.). On the left the situation where source and target distributions are the same. One the right the situation where target clusters are completely redrawn.

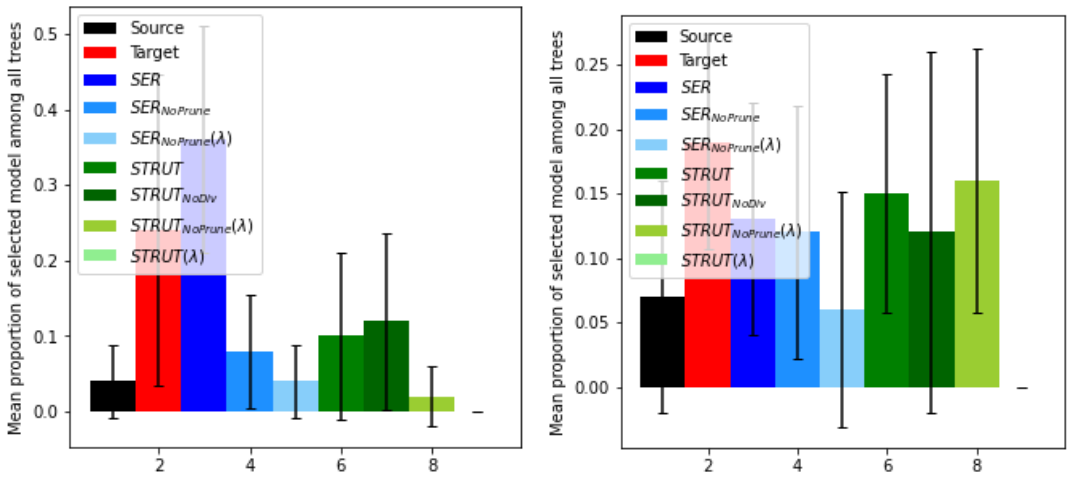


Figure 3.18: Visualization of STRF selection on synthetic data (situations 3. and 4.). On the left the situation where one third of source clusters are redrawn to obtain target distribution. One the right the situation where target clusters are translated versions of source clusters.

STRF interpretation on synthetic data : The first group of experiments concerns synthetic data using the Gaussian cluster generator previously presented. Four situations have been tested with 10 repetitions, each one initialized by a distribution of 15 clusters for each label in dimension 3 and with a source random forest of size 10 trained on 100 instances coming from this distribution. Then the target domain is created these rules for each of the 4 situations:

1. Target domain is exactly the same as source domain and target data are simply re-drawn from the same source distribution. In this situation there is no need of transfer and source model should remain the same.

2. Every cluster is randomly reset with new parameters, meaning target domain corresponds to a completely new distribution with no relation with source domain. In this situation transfer should be useless and the model needs a full new training instead.

3. One third of all the clusters are reset with new parameters and the rest stays unchanged, which is equivalent to deleting 5 clusters for each label and adding 5 new ones.

4. Each cluster's mean is translated on one axis with a random value of maximum one fifth of feature space bound.

Figure 3.17 corresponds to the 2 first situations (1. on the left and 2. on the right) and as expected STRF chooses respectively source and target model in majority, showing that the algorithm is able to indicate while no transfer is really needed while keeping a consistent model close to the source one or the target one depending on the situation. It can also be noted that among tested transfer algorithms, *SER* is the most selected one in these kinds of situations. It can be explained by the important flexibility of *SER* in terms of structure changes, as it is able to either keep decision trees unchanged or to adapt to a full new training by *extension* and *reduction* operations.

Figure 3.18 represents cluster addition/deletion situation and translations (3. on the left and 4. on the right). As we can observe *SER* is mostly selected by STRF in situation 3. whereas *STRUT* is preferably selected compared to *SER* in situation 4.. This confirms tendencies observed on unique decision trees in Figure 3.6 and shows again how STRF can bring information about the nature of the differences between source and target domains. Target decision tree being largely chosen in situation 4. might be interpreted as the fact that this situation is not very challenging compared to the cluster addition/deletion and a simple re-training may be sufficient.

STRF interpretation on fall data : As the transfer between simulated and real fall data is concerned STRF algorithm seems to select a small majority of *SER* trees while still using all of the variants of transfer algorithms, as well as source and target decision trees. We can assume that some part of target data are comparable to source data, showing source model utility, whereas others can be viewed as a completely new component of the distribution. Overall it may suggest not surprisingly that target data are richer in terms of events and need deeper decision trees.

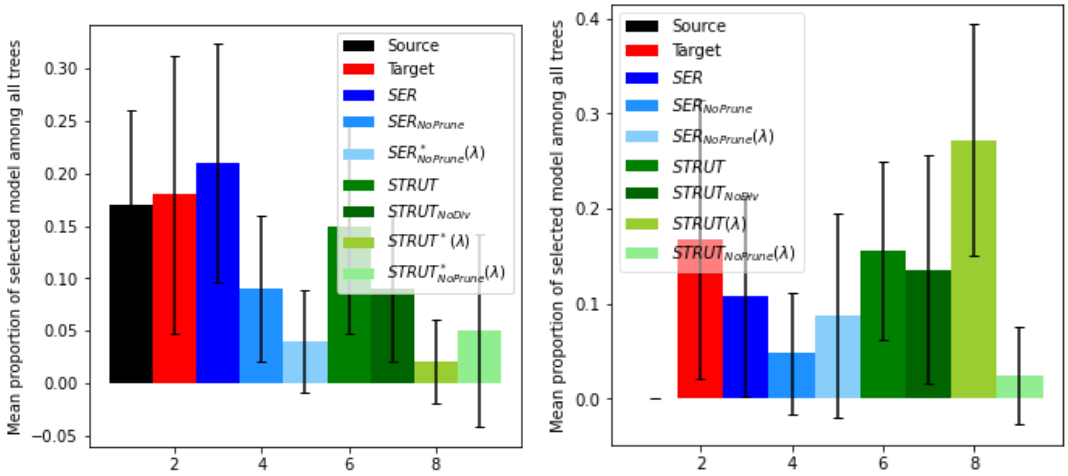


Figure 3.19: Visualization of STRF selection on fall detection data.

7 Conclusion

This chapter aims at tackling the differences between simulated falls data and real falls data using transfer learning to integrate real data into a random forest previously trained on simulated data. Based on a seminal work about model-based transfer on decision trees [216] two algorithms are tested, namely *SER* and *STRUT*, and we explain how class imbalance and data rarity can severely impact transfer learning. Indeed in practice we can collect almost as much real non-fall events as desired but real falls are still rare events. We illustrate this issue on *SER* and *STRUT* using our fall detection data and synthetic data and point out the two main mechanisms involved in these consequences : the *divergence gain* optimization and the pruning.

To solve this problem, we studied several variants of these algorithms and propose some generic adaptations. The key intuition of these adaptations consists in reducing

the minority class *pruning risk* while modifying decision trees, since this risk appears to have a significant impact over the original methods, decreasing performances and even leading to negative transfer. Our approach consists in considering the simplest class imbalance situation in target domain and formulate the *homogeneous imbalance* assumption. Based on this assumption, the *divergence gain* optimization can be redefined and the *pruning risk* can be estimated on each leaf of decision trees. The proposed adaptations $SER_{NP}(\lambda)$ and $STRUT_{NP}(\lambda)$ can be viewed as generalizations of the original algorithms as they depend on the parameter λ that quantifies class imbalance amplitude. The more target class proportions are close to source ones, the more λ_k values (with k the class index) are close to 1 and the more these adaptations are close to original $SER/STRUT$.

7.1 Class imbalance, pruning risk and assumptions about domain changes

This chapter shows the relevance of estimating the *pruning risk* while performing transfer learning on decision trees with methods involving pruning in target class imbalance conditions, even if the *homogeneous imbalance* assumption is not satisfied in practice. This estimation is done on each decision tree leaf individually however the real risk is not to lose one leaf in particular but rather to lose "too much" minority class leaves with regards to the distribution complexity of the minority class. Even though the right amount of these minority class leaves in the model is hard to quantify without any further assumption, a more general *pruning risk* could be defined as the probability estimate of pruning a precise number of leaves out of a decision tree. This kind of estimate might be hard to compute but the *homogeneous imbalance* assumption is sufficient for it to be well defined. So this perspective could lead to interesting development, such as the assessment of the mean expected amount of minority class leaf lost by pruning, and therefore more general strategies while transferring decision trees under class imbalance.

The *homogeneous imbalance* assumption shows how much modeling distribution changes between source and target can be useful to come up with some helpful indicators for undertaking adapted approaches. For example source/target transformations leading to class imbalance could be seen as a composition of a fixed balance transformation with a *homogeneous imbalance* transformation (which is revertible). Assumptions can also be done on source and target distributions themselves. For instance if data are supposed to come from a mixture of Gaussian clusters (like generated synthetic data presented earlier) then possible transformations between source and target can be decomposed into cluster addition deletion, clusters re-weighting and changes of cluster mean and covariance matrix parameters.

7.2 Selective transferred random forests (STRF) and heterogeneous transfer

In the context of STRF algorithm, there are two advantages in decomposing transformations between source and target distributions into different families of possible transformations according to particular assumptions about the shape of distributions. It permits to design particularized and complementary decision tree based transfer procedures associated with each family of transformations. Our intuition is that by including these complementary procedures, STRF must be able to handle any combination of

these transformations. This is the original idea behind *SER* and *STRUT*, with one algorithm supposed to cope with cluster addition/deletion type of transformation and the other assumed to cope with cluster squeezes/stretches and translations. Moreover, as illustrated in section 6, STRF selection would serve as a complete description of the transformation between source and target domains, which is highly valuable. Thus a natural extension of STRF would be to add other transformation based algorithms depending on some assumptions about distribution changes between source and target domains.

Finally this work focuses *homogeneous transfer* learning (meaning $(\mathcal{X}^S, \mathcal{Y}^S) = (\mathcal{X}^T, \mathcal{Y}^T)$) and its extension to *heterogeneous transfer* would be another interesting perspective. For that model-based transfer algorithms applied on decision trees need to be able to deal with feature space (when $\mathcal{X}^S \neq \mathcal{X}^T$) changes and/or label space change (when $\mathcal{Y}^S \neq \mathcal{Y}^T$) from source to target. While $\mathcal{Y}^S \neq \mathcal{Y}^T$, it concerns values of decision tree leaves and how they are assigned. As both *SER* and *STRUT* algorithms relabel tree leaves according to target labeled data, they could in theory still work in the context of different label spaces. However if *expansion* and *reduction* operations in *SER* can easily be extended in this context, it is not the case for the *divergence gain* optimization in *STRUT* that strongly depends on labels distribution at each node.

Considering the feature spaces change context where $\mathcal{X}^S \neq \mathcal{X}^T$, the objective can be either to add new features to the model or to remove some features (or both). Adding new feature is possible with *SER* but only on the expansion phase whereas not feasible in *STRUT* as it does not use any operation involving feature change. Removing features from the model remains the most difficult situation with regards to *heterogeneous transfer*. In the next chapter we define the notion of decision tree *equivalence* that could be helpful for this purpose and we explain this perspective in section 5. Thus by redefining some of the *SER* and *STRUT* operations and by integrating other ones to deal with label space or feature space changes, designing a STRF algorithm that can work in the *heterogeneous transfer* should be achievable.

4

Budgeted learning : industrial scaling up of decision tree models for embedded systems

1 Introduction

1.1 Machine learning subject to resource constraints

The previous chapter described several predictive models meant for daily activity monitoring of elderly, from feature set design to domain model transfer, without considering any limit in the amount and complexity of computations. Nevertheless these predictive models have to be implemented in an embedded system and to work with high reactivity, at least for fall detection. Thus the computation time of these models, as well as their size in the local memory of embedded system are constrained. These kinds of constraints that are related to some practical resources (in this case time and memory) are often referred to as "*budgets*" as they can not be exceeded and methods taking them into account in model design belong to the *budgeted learning* field.

Although such resource constraints cannot be neglected for the final application, keeping a non-limited in experimental context is still useful for assessing the most empirical accuracy obtainable, understanding real conditions prediction errors and for more detailed interpretation in general. In this chapter we study how to get a "simplified" version of complex models, using budgeted learning, in order to cope with embedded technology constraints, while keeping good predictive efficiency. Budgeted learning is part of methods that do not have only the loss function expectation as the objective function to optimize. It defines another objective function representing some costs during the training or the prediction of the model with some bounds on these costs that can not be exceeded (called budget).

Various budget learning situations

One major success of machine learning is its today applicability in a wide-range of industrial systems in numerous fields like IoT, security, healthcare or finance. In this context, the prediction model complexity is often adjusted through hyper-parameter tuning in the learning process, without taking deeply into account the capacity of the device intended to run the model. Indeed several variables, involving real resource consumption necessary for building a model and/or making it work in a limited device, are not directly included in classical machine learning, whereas these real world applications need at least computational resources for training or computing their outputs and it can become the main challenge of these applications. These resources are often subject to practical constraints and taking them into account during the learning and/or prediction phase is the purpose of budget learning. Moreover the rise of deep learning illustrates well that machine learning models tend on one hand to be more and more complex in term of size and inner computations, and on the other hand they are increasingly applied for real time predictions in systems that are more and more compact, autonomous and portable. Thus, budgeted learning is a vast and real concern for modern ground applications of machine learning. At the same time, the complexity of machine learning models is growing so as the scale of their application in real world, and the hard-wares they are implemented in are more and more portable and small [123], our application being a great example of this ambivalence [174]. Numerous recent works involve this topics on various kinds of models and considering different resource constraint frameworks. Some of them refer directly to industrial applications [163, 240], whereas others focus more on the theoretical framework of budgeted optimization

[95, 107].

In this context, considered resource constraints can be linked with data gathering, features computation, model size, time or energy consumption. For instance some of them embed models in robotic units which have access to too much observational inputs to concurrently treat all of them [65, 163], implying data acquisition costs. Another concerned application field is advanced real-time video processing with relatively "heavy" models [135, 165] where constraints can be linked to the model size, feature computations and prediction time. Concerning activity recognition [198] proposes a method alternating several wearable sensor to optimize the trade-off between accuracy and energy consumption.

So the nature of budgeted resources varies a lot depending on applications, and besides, these resources can be critical either in the *training* or the *prediction* phase [45, 93]. For example [44] investigates the question of costly acquisition of data attributes under a budget enumerating three main frameworks : *local budget constraint*, *global budget constraint* and *prediction on a budget*. The two first ones concern the training phase with a budget corresponding to a maximum number of attributes that can be acquired, either on each training instance [19] (local budget), or on overall training data set [75, 134] (global budget). In the third one attribute acquisition is also costly but in the prediction phase while the model is already trained [104]. In this budgeted frameworks comparison context, [44] comes up with theoretical results suggesting that predicting on a budget may be a harder task than training on a budget.

Budgeted active learning methods focus on budgets defined on training instances, whereas other approaches consider budgeted computation time of prediction and even both can be considered at the same time, for instance, this is the case for budgeted reinforcement learning [59]. In our case, we assume that the training is done under unrestricted conditions, with hypothetically unlimited resources. Conversely, as our model is intended to be embedded in a small electronic device, the cpu-time consumption is extremely constrained in the prediction phase occurring in real conditions.

1.2 Budget on the training phase

The problem of selecting a good subset of training data is known as *active learning* and presents an increasing interest nowadays due to the multiplication of technological means to gather data. While this data gathering is costly, whether costs are on raw observations, some attributes or labels, then budget learning approaches can be considered. For example numerous real applications use several correlated sensors but aim at restricting the amount of used one data dependently to be cost efficient [65, 163]. As any *optimal stopping* problem is close to a budgeted optimization a common approach to solve these kinds of sensors subset selection is to use variants or extensions of the *secretary problem* [13]. [176] uses a comparable method in the context of federated learning for the selection of IoT clients where communications between them are costly.

Concerning costly data gathering context, a particular case active learning under a budget focuses on the cost of labeling. For example [107] designs a selective sub-sampling strategy in the context of linear regression based on out-of-sample error bounds of Laplacian regularized least squares, in order to find an optimal subset of samples to label of budgeted size. In the context of online learning on data streams, [280] proposes an algorithm for online learning framework with imbalanced data and

asymmetric miss-classification costs, where the number of queries to obtain sample labels is budgeted.

Another field where several bridges with budgeted learning has been proposed is the *multi-armed bandit* problem (MAB), in the context of costly data exploration. Indeed most of the time MAB real application implies some limits on the exploration phase like time spent, number of trials, money or any kind of resource needed to pull an arm, or even the amount of arms that can be simultaneously pulled in the multi-player MAB framewok [31]. For example [101, 164] isolate a finite exploration period to estimate the best strategy where each experiment has a particular cost and the sum of them can not exceed a budget before accessing the exploitation phase. [74, 75] studies the attributes global budget situation, developing the parallel between MAB framework by assimilating each arm to a data attribute, which allows to use MAB ideas and algorithms to solve this problem. A recent innovative work in the MAB context introduces time costs in the reward itself to study optimal time allocation situations [32].

1.3 Budget on the prediction phase

Another budgeted learning framework is while training dataset is already acquired with a vast pool of usable features and costs are related to the model computations. Real-time application that have to apply model computations within a certain time are particularly concerned, which is then often referred to as budgeted *prediction time* [184, 247] or *test time* [182].

One family of methods to control prediction time consists in sequential computation models that adapt themselves to instances complexity or targeted prediction accuracy. For example [133, 135] define data dependent policies to sequentially acquire features based on their costs and the expected gain in accuracy they bring. In some cases features are directly coming from particular sensors and this kind of sequential acquisition approach results in concrete actions in the observational environment [234, 247]. By considering a low-cost model prediction as an observation, this sequential acquisition information principle can also be applied to sub-model evaluation while using ensemble models as it is formalized in [95], leading to some cascade association of sub-models [54, 262] or more generally in gating strategy [182] among these sub-models.

To sum up the task of building budgeted predictive models is a way of dealing with a trade-off between prediction accuracy and models costs and as developed in [95] and [133], sequential computations can be formulated as a set of actions leading to some *rewards* representing this trade-off. Thus, whether it is formulated as an *online learning* [163, 176, 280] or *reinforcement learning* problem [59, 133, 165, 198, 240], it may also happen that some resources are costly both in training and prediction phase and particularly when these phases are not distinctly separated and the model learns continuously based on new observations.

1.4 Budgeted learning on decision trees

So the first point while having an overall look of this topic is that one major categorization among budgeted learning situations, as presented by the authors in [45, 59], is whether the resource costs appear in the training or the prediction phase of a machine

learning model. Then considering budgeted *prediction time* learning, approaches based on sequential computations are particularly common and convenient. From these perspective decision tree based models appear to be particularly suited in this context as their are inherently sequential cascade models through their structure and they allow to extract sub-predictions quite simply.

Several recent works on budgeted learning with decision trees show the increasing interest in finding new ways to deal with real-world resource constraints in the context of models based on decision trees. Some methods incorporate directly budget constraints in a tree building algorithm to learn from scratch a budget sensitive model [54, 183] whereas others are designed to alter a pre-trained decision trees based model to make it fit some budget constraints [184].

One common approach while dealing with prediction time costs on decision tree based models is to include a penalization to these costs directly in the greedy building operations of decision trees. This can be done by adapting boosting trees methods, as the *Greedy-Miser* algorithm [262], that adds sequentially a new decision tree in a step-wise regression way at each step of the algorithm. Each new decision tree is built using modified version of CART with a budget sensible purity function which takes into account both empirical error and feature usage with a regularization parameter. In [264], authors present several boosting algorithms on decision trees for budget learning and extends Greedy-Miser for unlabeled data situation, performing then semi-supervised budgeted learning, and name their algorithm *Gradient Regularized Budgeted Boosting* (GRBB). Another local approach is proposed in [183], for inducing budget sensitive ensemble of decision trees, where the authors define a family of budget sensitive purity functions (class of admissible impurity functions) that can be used to greedily construct decision trees.

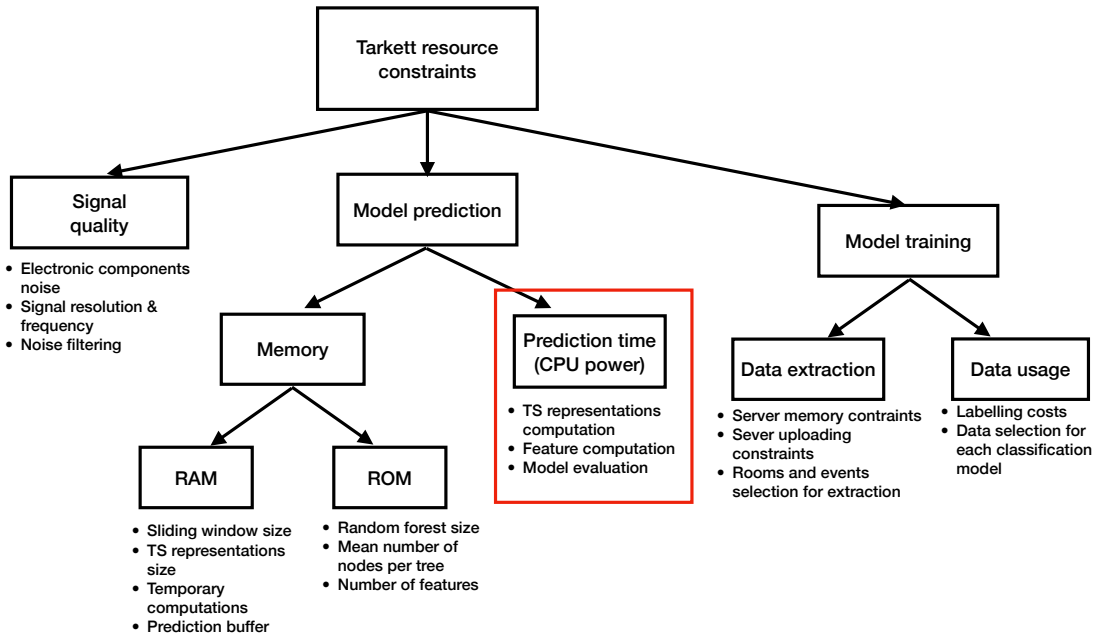


Figure 4.1: Possible use cases of budgeted learning for Tarkett monitoring application. This chapter focuses on the constrained prediction time situation, represented by the red rectangle.

In [184], the same authors propose an alternative algorithm with a more global optimization approach. The budget learning algorithm starts from random forest of pre-trained decision trees and then prunes them under budget constraints. They propose to solve a linear program corresponding to the best pruning combination on decision trees of a random forest, relatively to a dataset and feature acquisition costs, according to the trade-off between prediction accuracy and prediction computation time.

2 Problem statement : budgeted prediction time on random forests

2.1 Introduction on decision trees computation costs

In real-world machine learning application systems, main technological limitations in terms of computational resources can be divided in three groups :

- non-volatile or permanent memory B_v(used to store information permanently like model parameters);
- volatile memory B_{nv}(used to make all the intermediate computations needed by the model);
- computing power v_{comp} which determines computation speed (measured as an amount of basic operation the system can achieve within a time unit).

In a nutshell non-volatile memory restricts *model size* whereas volatile memory and computing power limits the amount of data processing, feature computations, model evaluation and then the *prediction time*. For technological reasons, nowadays most of computing devices can afford as much non-volatile memory as needed for a machine learning model, contrary to volatile memory and computing power that are usually more valuable and critical industrial aspects. But in some cases where the parameter space of the model can be huge [240] non-volatile memory limitation can still matter.

To sum up, while aiming at embedding a predictive model in a resource limited computing device, denoting volatile memory B_v, non-volatile memory B_{nv} and computing power v_{comp}, several characteristics of the model are budgeted. Model size has to be lower than B_v, temporary computations have to take less memory than B_{nv} at anytime and the prediction time is equal to the total amount of needed computations multiplied by v_{comp}. As specified, for our application we only consider prediction time budget which relies on two types of computations: those used to compute features from the original data and the internal computations of the model itself assuming needed features are already accessible. This distinction is quite general among machine learning models and the associated costs to each of these computations are referred to as *feature acquisition cost* and *evaluation cost*. Nevertheless the context of decision tree is particularly convenient as every internal computation is the same simple operation (an equality test between two values), meaning a constant cost for each node.

For our application we assume that we have enough permanent memory to store our random forest models. On the other hand signal representations and features computations need to be stored in the volatile memory and above all these computations

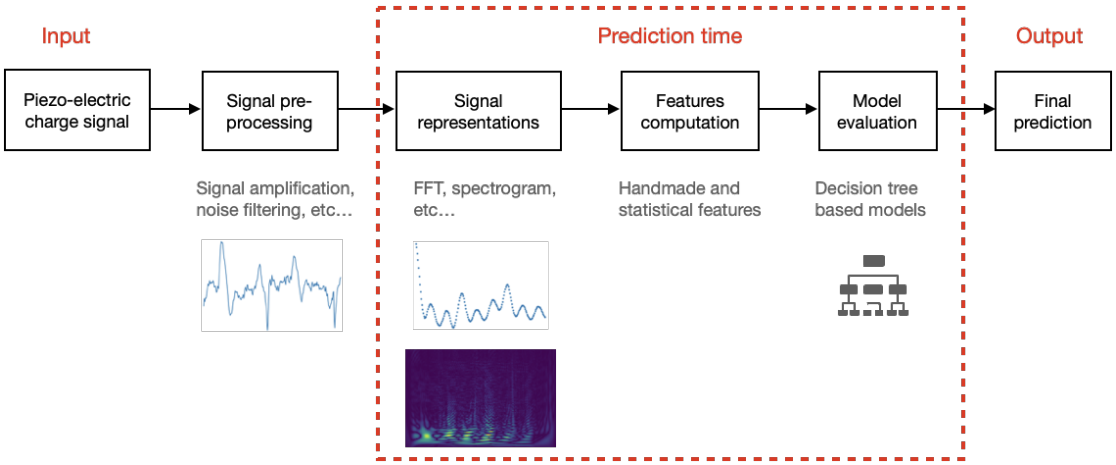


Figure 4.2: Computation steps for monitoring detection tasks from sensor to prediction

have to be done in a limited time as we aim at achieving real time detections. For that, the computing power is the critical parameter of the embedded system and it directly implies a budget on the prediction time. More concretely, with a desired prediction frequency f_p (ideally $100Hz$ in our case), the whole model must be computed in less than $\frac{v_{comp}}{f_p}$ operations for every input or at least in mean. Figure 4.2 represents all the different computation steps from the initial sensor input to the final model prediction. The three main time consuming parts are framed in red and refer to the time spent to compute signal representations like auto-correlation or spectrogram, the time to compute features based on these representations and the time to evaluate the model once all the needed features are computed.

Regardless of the type of model and application these computation time costs can be sorted into two main costs: a *feature acquisition* cost C_{fa} and a *model evaluation* cost C_{ev}. In general these costs can depend both on the model $\mathcal{M}$ and a data input x such that the total prediction time cost is defined by:

$$C(\mathcal{M},x) = C_{fa}(\mathcal{M},x) + C_{ev}(\mathcal{M},x). \tag{4.1}$$

In the next sections we describe how to evaluate these two costs on decision trees and random forests for our application and based on time consumption parameters presented in Table 4.1.

Computation	Parameters	Dimension	Type of cost
Signal representations	$(g_i)_{1\leq i\leq n_g}$	$n_g = 6$	C_{fa}
Features	$(c_j)_{1\leq j\leq d}$	$d = 128$	C_{fa}
Node splits	k_{ev}	1	C_{ev}

Table 4.1: Computation time parameters involved in the total prediction time

2.2 Cost definitions : feature acquisition and evaluation cost

For any $s \in \mathbb{R}^N$ truncated discrete signal of size N a pool of d features $(\phi_1,...,\phi_d)$ can be computed on s, then the feature space sample corresponding to s is the vector

$(\phi_1(s),...,\phi_d(s))$ and is denoted x_s and later simply x to simplify the reading. Let's consider a machine learning model $\mathcal{M}$ (in our case a pre-trained decision tree or random forest) and a dataset of n labeled samples $(X,Y) = \left((x^{(1)},y^{(1)}),...(x^{(n)},y^{(n)})\right)$ with a feature space of dimension d. We assume first that each feature ϕ_j (with $1 \leq j \leq d$) has an individual *feature acquisition cost* c_j (time to compute ϕ_j) and an *evaluation cost* k_{ev} representing the time needed to assess each internal node split, assuming it is the same for every node.

Feature acquisition cost (C_{fa}) : For a given decision tree T, any sample $x_s \in \mathcal{X}$ is associated with a precise path leading to a leaf, which is a sequence of nodes each composed by a feature and a threshold on this feature. Then a *feature usage* vector $\Phi_T[x] \in \mathbb{N}^d$ can be defined indicating how many times each feature is used in this path. We denote also the vector $\Phi_T^{bin}[x_s] \in \{0,1\}^d$ indicating for each feature whether it is used or not in the path of x_s in T.

Then the computation time spent to compute features needed in the path of x_s indicated by the ones of the vector $\Phi_T^{bin}[x_s]$. Then the feature acquisition cost is defined as $C_{fa}(T,x_s) = \sum_{j=1}^{d} c_j \Phi_T^{bin}[x_s]_j$.

While dealing with an ensemble of decision trees for a boosting model or a random forest any feature needs to be computed only once for all the trees using it. So for a random forest Rf, the union of all the different paths of x_s in the trees of the forest can be considered, defining then in the same way vectors Φ_{Rf} and Φ_{Rf}^{bin} that describe the feature usage for the random forest prediction of x_s. It means that for a given featureϕ_j, $\Phi_{Rf}^{bin}[x_s]_j = 1$ if and only if there is at least one tree T of the random forest where $\Phi_T^{bin}[x_s]_j = 1$. Similarly the feature acquisition cost of the random forest is $C_{fa}(Rf,x_s) = \sum_{j=1}^{d} c_j \Phi_{Rf}^{bin}[x_s]_j$.

Evaluation cost (C_{ev}) : For a decision tree T it corresponds to the computation time spent to assess all the node on the path of x_s in T, having already computed needed features. Then by denoting $L_T(x_s)$ the length of this path the evaluation cost is proportional this length: $C_{ev}(T,x_s) = k_{ev} L_T(x_s)$. According to the definition of the *feature usage* vector, this length is equal to the L^1-norm of $\Phi_T[x_s]$ such that: $C_{ev}(T,x_s) = k_{ev}\|\Phi_T[x_s]\|_1$.

For a random forest Rf this cost is proportional to the cumulative length of all paths of x_s in the different trees:

$$C_{ev}(Rf,x_s) = k_{ev}\|\Phi_{Rf}[x_s]\|_1 = k_{ev}\|\sum_{T\in Rf}\Phi_T[x_s]\|_1 = k_{ev}\sum_{T\in Rf}\|\Phi_T[x_s]\|_1$$

Finally with $\mathcal{M}$ a decision tree or an ensemble of decision tree model, for a given sample x, the total *prediction time cost* of $\mathcal{M}$ is defined as :

$$C(\mathcal{M},x) = C_{fa}(\mathcal{M},x) + C_{ev}(\mathcal{M},x) = \sum_{j=1}^{d} c_j \Phi_{\mathcal{M}}^{bin}[x]_j + k_{ev}\|\Phi_{\mathcal{M}}[x]\|_1. \tag{4.2}$$

It is interesting to note that the cost function $C(\mathcal{M}, x)$ depends on the structure of $\mathcal{M}$, through the feature usage vector $\Phi_{\mathcal{M}}[x]$, but not very on the associated decision function. Indeed, contrary to the loss function that fully depends on leaves labels, the *prediction time cost* function is independent from these labels. Moreover we can directly observe from these definitions that the *evaluation cost* of a random forest is equal to the sum of evaluation costs of all its trees, whereas the *feature acquisition* cost of a random forest is always lower than the sum of feature acquisition costs of its trees, making C_{fa} sub-modular with regards to the set of considered decision trees.

$$C_{ev}(RF, x) = k_{ev} \sum_{T \in RF} \|\Phi_T[x]\|_1 = \sum_{T \in RF} C_{ev}(T, x) \tag{4.3}$$

$$C_{fa}(RF, x) = \sum_{j=1}^{d} c_j \mathbb{1}_{\{\sum_{T \in RF} \Phi_T^{bin}[x]_j) > 0\}} \leq \sum_{T \in RF} C_{fa}(T, x) \tag{4.4}$$

2.3 Specific cost definitions and signal representation costs

Several specific cases can be derived from this general definitions depending on particularities of each practical resource constrained situation. These different situations can be summarized into three main questions:

- Is the *evaluation cost* C_{ev} negligible compared to the *feature acquisition cost* C_{fa}?
- Do features need to be all computed before applying the model or can they be computed "on-demand"?
- Is the budget defined on the "worst-case" scenario or on the mean expected computation cost?

Negligible evaluation acquisition cost : Frequently feature costs are large and/or model size is relatively low. In this case the computation cost is simplified as $C(\mathcal{M}, x) \approx C_{fa}(\mathcal{M}, x)$. For example [183] considers this framework for random forests meaning that depths or the amount of trees are assumed to not impact on computation time which only depends on used features. This approximation about computation costs can also be encountered on other types of model like in [59] with a budgeted reinforcement learning model using a sequential feature acquisition random policy. Other simplification of the *evaluation cost* are possible, for instance Greedy-Miser[263] considers this cost constant for each tree (n other words the depth of trees is considered constant), meaning that the total *evaluation cost* of the boosted tree ensemble model is directly proportional to the number of trees.

Data dependence of feature computations : There are mainly two ways of managing feature computations for decision trees: one is simpler to implement and the other is more efficient computationally. The simplest way consists in computing every feature present in the model first and then apply decision trees. Then the *feature acquisition cost* is the same for every inputs: $C_{fa}(\mathcal{M}, x) = C_{fa}(\mathcal{M})$, this framework is considered for example in Greedy-Miser and GRBB [263, 264]. On the contrary features can be computed "on-demand" at each node of decision trees making the *feature acquisition*

cost depending on inputs and paths they follow. In other words, this distinction results in whether or not the computation cost C is considered *data dependent*. For instance some budgeted boosting models like Speed-Boost[106] and CEBG[196] or budgeted algorithms on random forests [183, 184] consider a data dependent cost definition. More generally in budgeted learning, data dependent computation costs can also be found with other kinds of decision tree inspired models[54, 262] or even completely different models [182, 248].

Bounding mean cost or worst-case cost : Depending on real conditions of the prediction problem model outputs can be of no use if the prediction time exceed the critical budget constraint and in this case it is the *"worst-case" cost* $\max_{x \in \mathcal{X}} \{C(\mathcal{M}, x)\}$ that is strictly constrained, as considered in [263, 264]. In other situations the prediction time budget is a softer constraint that can be exceeded exceptionally if respected in average. For example most of the time in a real-time application with continuous predictions, what really matters is the *mean expected cost* $\mathbb{E}[C(\mathcal{M}, x)]$, as defined in [106, 184], in order to not have too much delay with data input stream. Of course this distinction only matters if C definition is data dependent, as if not the maximum cost and the mean expected cost are equal.

Table 4.2 summarizes all these particularities in cost function definition and budgeted problem formulation for decision tree based models but it is noticeable that all these variations can also be encountered for other kinds of budgeted models and have to be chosen depending on the application. For instance a recent work studies the budgeted learning problem with heavy neural network architectures on data independent and budgeted maximum cost framework, with budgets both on prediction time and memory storage [240]. Moreover no further assumption on C definition is required by this method except that it can be measured during training, implying that it is usable with any definition of C.

With a particular concern for flexibility and scalability, our work on budgeted learning firstly introduces in the next section an adapted C cost function definition on random forests considering *signal representation* computations, and then proposes a general genetic algorithm working with any definition of C and that can be extended to several concurrent budget constraints.

Grouped features and shared computation costs

Previous *feature acquisition* cost definition considers that each feature has a fixed cost independently to other features. Nevertheless some real implementations use more complex and intricate feature computations. For instance a common situation is while some groups of features are not directly computed from raw data but rather on previous data processing computations, which are in general computed on every instance. It is exactly our case as we consider groups of feature defined on several signal representations: raw signal, derivative, integral, autocorrelation, Fourier transform and spectrogram. Each of these signal representation corresponds to different computation cost and including a feature in a model also implies to invest in computing its associated signal representation. This kind of "feature groups" with shared preliminary computations is frequent while features are computed in several steps from initial data or organized in

successive layers. [196] takes into account a comparable framework in *feature acquisition* cost definition with some features that need to be applied on all inputs at once and evokes separable convolution filters as an example. In our case it is a bit different as each proper feature needs its corresponding signal representation to be computed and this sort of computation dependency is frequent in real applications. For instance it is equivalent to consider several sensors having each their own acquisition cost for getting their measurements on which some groups of features can then be applied.

Each of these groups corresponds to additional computation costs and including a feature in a model also implies to invest in computing its associated *signal representations* as illustrated in Figure 4.2. We formalize this situation by introducing new costs g_k for each feature group acquisition (with n_g the number of groups and $1 \leq k \leq n_g$) and a matrix G of size $n_g \times d$, where $G_{i,j}$ indicates if feature f_j belongs to the i-th group. In this situation a feature group usage vector can be defined as $\phi^{bin}_{g,\mathcal{M}}[x] = \mathbb{1}_{\{G.\phi^{bin}_{\mathcal{M}}[x]>0\}}$.

Then the total *feature acquisition cost* becomes:

$$C_{fa}(\mathcal{M},x) = \sum_{k=1}^{n_g} g_k \phi^{bin}_{g,\mathcal{M}}[x]_k + \sum_{j=1}^{d} c_j \phi^{bin}_{\mathcal{M}}[x]_j. \tag{4.5}$$

2.4 Trade-off between the prediction error and the prediction time

The budgeted learning task can be formalized as a constrained minimization problem. Indeed, it aims at minimizing a mean empirical loss relatively to a loss function l(as in standard machine learning problems), but in a constrained space of models that have a prediction time bounded by a budget $B > 0$.

More formally, let $\mathcal{H}$ be a hypothesis space of classifiers from $\mathcal{Y}$ to $\mathcal{Y}$, in our case a space of models based on decision tree classifiers (e.g. random forests). For each model $\mathcal{M} \in \mathcal{H}$ corresponds a *decision function* $h_{\mathcal{M}}(.)$ from $\mathcal{X}$ to $\mathcal{Y}$ and a *computation cost* function $C(\mathcal{M},.)$ from $\mathcal{X}$ to $\mathbb{R}_+$. Moreover, let ℓ be a *loss function*, (X,Y) a couple of random variables following a distribution $\mathbb{P}_{XY}$ over $\mathcal{X} \times \mathcal{Y}$.

Then, if it exists, the optimal solution to the *budgeted prediction time* problem is:

$$\mathcal{M}^* = \underset{\mathcal{M}\in\mathcal{H}}{\operatorname{argmin}} \ \mathbb{E}_{XY}[\ell(h_{\mathcal{M}}(x),y)] \qquad \text{subject to } \mathbb{E}_{XY}[C(\mathcal{M},x)] \leq B \tag{4.6}$$

$$\text{or subject to } \max_{x\in\mathcal{X}}\{C(\mathcal{M},x)\} \leq B. \tag{4.7}$$

Assuming n i.i.d samples $(x^{(i)},y^{(i)})$ are drawn from the distribution $\mathbb{P}_{XY}$, $\mathcal{M}^*$ can be approached the solution $\hat{\mathcal{M}}$ of the associated empirical constrained minimization problem:

$$\hat{\mathcal{M}} = \underset{\mathcal{M}\in\mathcal{H}}{\operatorname{argmin}} \ \frac{1}{n}\sum_{i=1}^{n} \ell(h_{\mathcal{M}}(x^{(i)}),y^{(i)}) \qquad \text{subject to } \frac{1}{n}\sum_{i=1}^{n} C(\mathcal{M},x^{(i)}) \leq B \tag{4.8}$$

$$\text{or subject to } \max_{x\in\mathcal{X}}\{C(\mathcal{M},x)\} \leq B. \tag{4.9}$$

With either a constraint on the mean computation time(equation 4.7) or on the "worst-case" computation time (equation 4.7).

There are mainly three different families of methods to tackle this problem for budgeted learning on decision tree ensembles :

1. Local cost-sensitive optimization methods building decision trees by considering at each node an optimal trade-off between loss and cost function values.
2. Variants of usual gradient boosting approach on decision trees but replacing the loss function by a budget sensitive objective function.
3. Global cost-sensitive optimization methods by modifying pre-trained decision trees with regards to budget constraints.

Algorithm	**Max/Mean**	**F.A. cost**	**Ev. cost**	**Data dep**	**Family**
Feature budgeted RF [183]	Max	yes	no	yes	1
Speedboost [106]	Mean	yes	yes	yes	2
Greedy miser [263]	Max	yes	yes	no	2
CEGB [196]	Mean	yes	yes	yes	2
GRBB [264]	Max	yes	yes	no	2
Pruning on a budget [184]	Mean	yes	yes	yes	3

Table 4.2: Categorization of budgeted learning algorithms on decision trees

Each family of methods uses a form of trade-off between prediction accuracy and computation cost that intervenes at different levels during the training. The first category consists in building directly from scratch a budget aware model conserving the idea of greedy induction but altering the *impurity function* to include computation costs. For instance [183] keeps the bagging concept of random forests to train budget sensitive decision trees, using the ratio between feature cost and *impurity gain* to select every node split during tree induction. Authors define a family of *admissible impurity functions* and show theoretical guarantees of this method regarding the maximum features acquisition cost of induced trees.

The second family concerns gradient boosting strategies for decision tree ensemble induction. The main idea is to extend functional gradient descent used in boosting by replacing the usual *loss function* by a new objective function that takes into account computation cost. Thus [263] defines a lagrangian formulation of the trade-off between the *loss function* l and a relaxed version of the prediction time cost function C to extend gradient boosting strategy, known as Greedy-Miser. This idea has been exploited recently by [196] that applies the same approach with a slightly different definition of C and [264] that extends Greedy-Miser to the semi-supervised framework. SpeedBoost[106] is a comparable boosting algorithm defining the accuracy/computation cost trade-off as a ratio between accuracy gain and computation time spent. Moreover it proposes a cascade optimization of sub-classifiers to allow anytime prediction for flexible budgets.

Comparably to cascade arrangement of sub-classifiers for budgeted learning purpose [54, 106], the remaining family of method concerns global budget sensitive approaches (by opposition to local methods) on a pre-trained ensemble of decision trees. For example [184] proposes to solve the optimal pruning problem of a pre-trained random forest to fit a given budget. This category presents the advantage of providing a budgeted model that relies on a reference model with better accuracy that can be used in non-budgeted context. In an industrial context this can be highly desirable for maintenance, feed-backs and future improvements, this is why we chose this specific strategy. Nevertheless to be really efficient in terms of budgeted learning, these

approaches often need the reference model to be structurally suited for the global budgeted optimization. That is why for example authors in [184] advice to apply their budgeted pruning algorithm on a random forest previously trained by the local budgeted learning algorithm presented in Nan et al. [183]. To overcome this drawback we develop in the next sections a global budgeted learning algorithm on random forest based on genetic programming that uses the notion of *equivalent decision trees* to be efficient regardless of the initial structure of the reference random forest.

For our application we are mainly focused on the mean prediction time cost of our model but depending on the situation other kind of constraints on the cost function C can be added to this framework. Indeed it might be sometime interesting to study both budgeted maximum and mean costs concurrently or, as evoked earlier, budgets on temporary computations storage (RAM) or on the total model size (ROM). As an example in [240] authors only assume that cost is measurable during training and experiment their method for budgeted computation cost as well as for memory consumption cost for models based on combination of several neural networks.

Another remark concerns the distinction we choose to make between a model $\mathcal{M}$ and its decision function $h_{\mathcal{M}}$, which is motivated by the observation that in this context of budgeted prediction time two different models can have the same decision function with different costs as explained in details in section 3. A common way to address this constrained optimization 4.7 is to minimize the trade-off prediction error and prediction time by introducing and calibrating a parameter $\alpha > 0$:

$$\hat{\mathcal{M}} = \underset{\mathcal{M}\in\mathcal{H}}{\operatorname{argmin}} \sum_{i=1}^{n} \left(\ell(h_{\mathcal{M}}(x^{(i)}), y^{(i)}) + \alpha C(\mathcal{M}, x^{(i)}) \right). \tag{4.10}$$

This optimization problem is then precised for each situation depending on C cost function definition as detailed in previous section.

In our particular case we consider both *feature acquisition cost* and *evaluation cost*, as well as the possibility to take into account signal representation computation costs explained in section 2.3. Moreover we aim at performing real-time predictions following the signal frequency of $100Hz$. As presented in chapter 2 these predictions are then smoothed to provide the final detection task. So what matters the most is the mean prediction time the random forest model takes to treat one observation and it is not critical if the model spends more time on some observations as long as the induced delay is caught up on other observations. Nevertheless focusing on the mean prediction time requires a precise estimation of sample distribution over the feature space. Four distinct budgeted optimization frameworks are considered in this work for our embedded fall detection application, depending on whether the feature acquisition cost is considered data-dependent and if it takes into account signal representation computation costs. Then the general formulation of the budgeted prediction time learning problem can be rewritten in our situation :

$$\hat{\mathcal{M}} = \underset{\mathcal{M}\in\mathcal{H}}{\operatorname{argmin}} \sum_{i=1}^{n} \left(\ell(h_{\mathcal{M}}(x^{(i)}), y^{(i)}) + \alpha \left(\sum_{k=1}^{n_g} g_k \phi^{bin}_{g,\mathcal{M}}[x]_k + \sum_{j=1}^{d} c_j \Phi^{bin}_{\mathcal{M}}[x^{(i)}]_j + k_{ev} \|\Phi_{\mathcal{M}}[x^{(i)}]\|_1 \right) \right). \tag{4.11}$$

Thus the four different scenarios come from this 4.11 equation, where ignoring signal representation costs is the same as considering $g_k = 0$ for every k and considering

data-independent feature acquisition means that $\Phi^{bin}_{\mathcal{M}}[x^{(i)}] = \Phi^{bin}_{\mathcal{M}}$ is a constant vector of size d.

3 Equivalent decision trees for budget learning

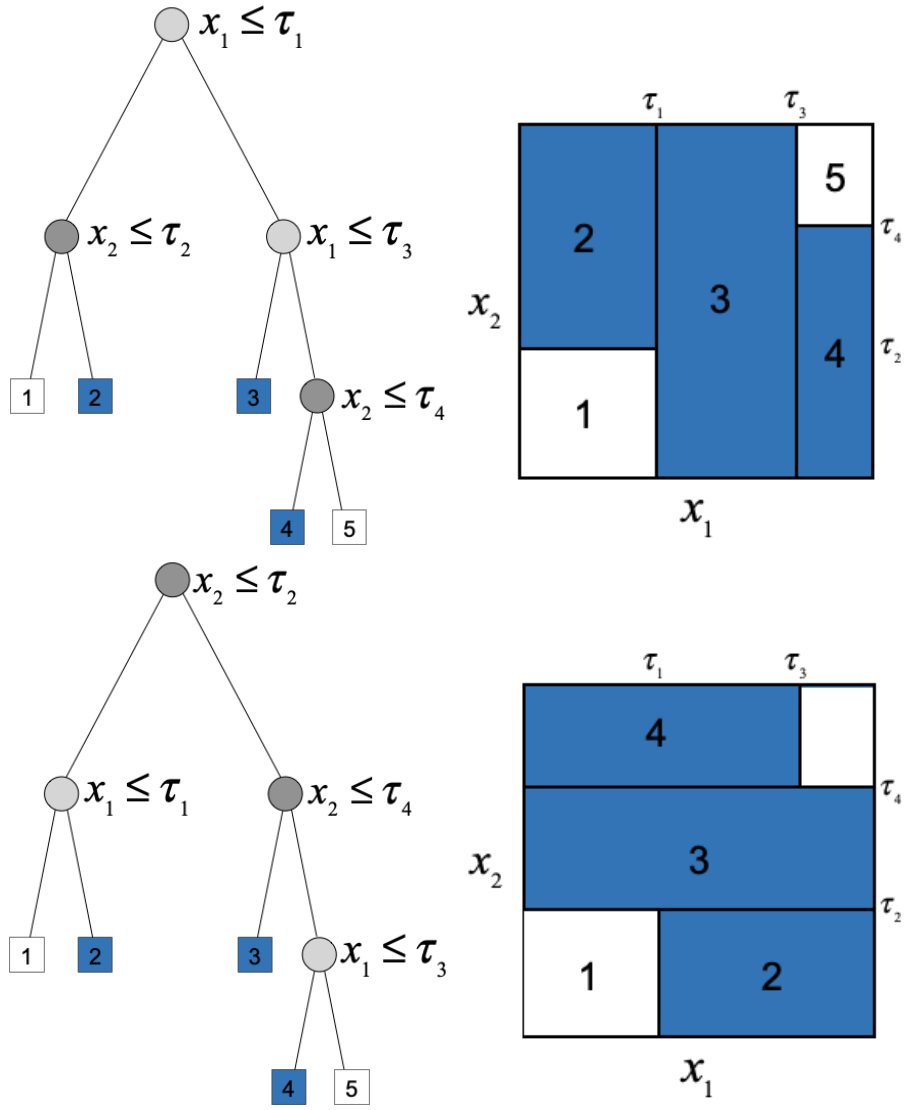


Figure 4.3: Illustration of 2 equivalent decision trees.
These 2 trees are structurally different, but by denoting *white* and *blue* colors as labels 0 and 1, they both share the same decision function h such as $h(x) = 0$ if $x_1 < \tau_1$ and $x_2 < \tau_2$ or if $x_1 > \tau_3$ and $x_2 > \tau_4$ (and then $h(x) = 1$ elsewhere).

Previous section shows that the prediction time of a decision tree T on a sample x depends on its structure through the feature usage vector $\Phi_T[x]$. Especially if feature computations are done "on-demand" along the path of x in T, the order of feature usage on this path impacts the prediction time. Thus in this section we suggest to distinguish between a decision tree structure and its associated decision function. Using the notion

of *equivalence* between decision trees, we show that two decision trees can share the same decision function while having different prediction time costs and propose to exploit this notion for a budgeted learning purpose.

As decision trees performance considers most of the time only classification accuracy assessed through a *loss function* that only depends on the *decision function*, the possibility of using equivalent decision trees is often ignored because it is irrelevant regarding the *loss function*. Nevertheless in transfer or budget learning a large variety of structural algorithms are applied on previously trained decision trees, meaning that the same algorithm might result in different outputs whether it is applied on a tree or on one of its equivalent trees. Moreover budgeted learning is particularly concerned by this notion of equivalence as it considers not only a *loss function* but also a *cost function* that highly depends on trees structure.

3.1 Equivalence definition

Before considering any associated classifier or regressor model, a decision tree is composed by an underlying rooted tree structure defined as an oriented graph. This structure can even be expressed independently to feature and label spaces $\mathcal{X}$ and $\mathcal{Y}$, and can play a major role in costs definition for budgeted learning.

Definition 4.1. *Set of binary trees*

Consider $\mathcal{B}_T$ the directed graphs set of non-empty and full binary tree (meaning that each node has either 0 or 2 children). For a given binary tree *$\mathcal{G}_T \in \mathcal{B}_T$ let's denote $\mathcal{L}(\mathcal{G}_T)$ the set of terminal vertexes (with 0 children) and call it the set of $\mathcal{G}_T$'s leaves.*

Definition 4.2. *Given a label space $\mathcal{Y}$ and a feature space $\mathcal{X}$ of dimension d and a binary tree graph $\mathcal{G}_T = (V_T, E_T)$, a decision tree can then be defined as :*

$$T: \quad V_T \to \{1,...,d\}, \mathcal{X} \cup \mathcal{Y}$$
$$v \mapsto \begin{cases} (i_v, \tau_v) \in \{1,...,d\}, \mathcal{X} & \text{if } v \in \mathcal{I}(\mathcal{G}_T) \\ y \in \mathcal{Y} & \text{if } v \in \mathcal{L}(\mathcal{G}_T) \end{cases}$$

For any leave $v_l \in \mathcal{L}(\mathcal{G}_T)$ there exists a unique sequence of nodes $(v_0,..,v_p)$ from the root v_0 to the direct predecessor v_p of v_l. Then v_l is associated with a subspace $X^l \subset \mathcal{X}$ of the form $X^l = \bigcap_{i=0}^{p} V_i$ with $V_i = \{x \in \mathcal{X} / x_{v_i} < \tau_{v_i}\}$ or $V_i = \{x \in \mathcal{X} / x_{v_i} > \tau_{v_i}\}$.

We will consider only consistent decision trees, meaning that X^l leaf subspaces are non-empty and forms a partition of $\mathcal{X}$. Then the decision function of T can be defined on each element of this partition as : $h_T(X^l) = T(v_l)$.

Definition 4.3. *Decision tree equivalence*

Let's denote $\mathcal{T}$ the set of decision trees from $\mathcal{X}$ to $\mathcal{Y}$ as defined on 4.2 and for any given decision tree T its decision function is denoted as h_T. Two decision trees $T_1, T_2 \in \mathcal{T}$ are equivalent *if and only if their decision functions are equal on all the feature space $\mathcal{X}$.*

$$T_1 \sim T_2 \iff h_{T_1} = h_{T_2} \iff \forall x \in \mathcal{X}, h_{T_1}(x) = h_{T_2}(x)$$

In other words labeled partitions induced by leaves of T_1 and T_2 are equivalent.

$$\forall (v_{l1}, v_{l2}) \in \mathcal{L}(\mathcal{G}_{T_1}) \times \mathcal{L}(\mathcal{G}_{T_2}), \quad X^{l1} \cap X^{l2} \neq \emptyset \implies T_1(l1) = T_2(l2) \tag{4.12}$$

Remark 4.1. *The equivalence characterization in Equation 4.12 is used in a recursive procedure presented in Section 3.3 to randomly generate equivalent decision trees.*

3.2 Related works

Since decision tree induction with CART algorithm was introduced for machine learning in 1984 [36], several computer science works have been done on the relations between decision trees and boolean algebra [57, 177]. Indeed decision functions associated with these models are tightly linked to boolean functions. To figure it out one can view a tree path leading to a leaf as a product of boolean variables corresponding to node split tests and the union of several paths as a boolean addition of these products. It is from that perspective that numerous computer science works investigated expression, simplification, optimization and properties of decision rules defined by decision trees [5, 40, 56, 111].

So in the same way that equivalence between boolean decision lists can be established, equivalence between decision trees has already been studied from this boolean algebra point of view [109, 275]. In particular once equivalence between decision trees is formulated, the question of optimality with regards to the complexity of the decision function expression arises, which is comparable to the boolean expression factorization problem. It has been proved that "optimal" decision tree (in terms of tree size) search is NP-complete [124, 276]. Nevertheless the exploration of heuristics for optimizing or simplifying decision tree rules, whether it deals with strict equivalence or not, is still a vast topic of research [5]. For example to avoid NP-completeness difficulty [56] investigates the sub-problem of equivalent tree search within the range of *reduced trees*, using the property that if an equivalence is found for a sub-tree then equivalence can be deduced for the total tree.

All these different works about decision tree rules are not particularly recent and take into account neither any statistical distribution nor any feature cost, but optimizing the structural complexity of decision trees can yet be seen as a particular case of budgeted learning as these models size might be *evaluation costs* on their own as defined earlier. Then the notion of "optimal" tree search intervening in these previous works could be generalized with extended tree costs definition developed in previous section, and could also incorporate data distribution information. In general for regression or classification problems, if statistical distributions are considered for scoring tree-based models, reducing tree structural complexity helps to avoid over-fitting [34, 139, 215] or may at least increase the interpretability (comprehensibility) of models [254]. So there are plenty of reasons motivating that in real-world applications studying the variability of possible tree structures for a given decision function, like through the notion of equivalence, can bring some improvements to machine learning applications, especially with regards to budgeted learning considerations.

3.3 Randomized equivalent decision trees generation

Being able to exploit the notion of equivalence between decision trees for budgeted learning requires first a way to generate equivalent decision trees. For that we designed a recursive randomized tree induction algorithm based on equivalence characterization described by equation 4.12, presented in pseudo-code in 4.1, using exactly the same

Algorithm 4.1 Randomized equivalent tree

procedure $RandEqRec(T, T_{eq}, path)$
  if $path = null$ **then**
    $T_{eq} \leftarrow NewTree()$
  end if
  $\phi, \tau \leftarrow CohSplits(T, path)$
  $C = h_T(path)$
  if $|C| > 1$ **then**
    // π : splits random drawing policy
    $\phi_k, \tau_k \leftarrow ChooseNewSplit(\phi, \tau, \pi)$
    $T_{eq} \leftarrow NewNode(\phi_k, \tau_k, \pi)$
    $path_l, path_r \leftarrow Childs(T_{eq}, path)$
    $T_{eq} \leftarrow RandEqRec(T, T_{eq}, path_l)$
    $T_{eq} \leftarrow RandEqRec(T, T_{eq}, path_r)$
  else
    $C = \{c_i\}$
    $T_{eq} \leftarrow NewLeaf(c_i, T_{eq}, path)$
  end if
  return T_{eq}
end procedure

splits as the original decision tree but in a different order. The idea behind that is to produce structural variety by equivalent tree generation in order to allow more possibilities in terms of computation cost reduction,while still keeping the initial decision function.

The principle is to compare during the induction any new path with the leaves of the original tree having intersection with this path. If only one class is represented by these leaves a new leaf of this class can be created on the newly generated tree ensuring the equality of the decision function on this leaf. On the contrary, if several classes are represented in the intersection between a path and original tree leaves, then the algorithm continue to add splits to this path.

Splits selection during the induction of the randomized equivalent tree relies on two functions referred as *CohSplits* and *ChooseNewSplit*. *CohSplits* assesses among all splits of the original tree which are consistent with the current path, meaning whether a split is out of the subspace of $\mathcal{X}$ associated with the current path or not. Then *ChooseNewSplit* selects, randomly with a policy π one split to add to the current path from the previously verified consistent splits. To keep the algorithm simple and the splits order fully random we choose for π a uniform policy over all the consistent split candidates but a wide variety of policies can be used. For instance one might prefer a policy linked to the feature importance to increase the probability to get smaller equivalent trees or to take directly into account in this policy feature acquisition costs.

As the set of all splits of the original tree is finite and the induction process considers every consistent split for each new node, it ensures that any induced path will have at a certain point a unique class intersection relatively to the original decision tree, entailing that this algorithm will end up necessarily with an equivalent decision tree. Nevertheless it is important to note that the heaviest a decision tree is in terms of

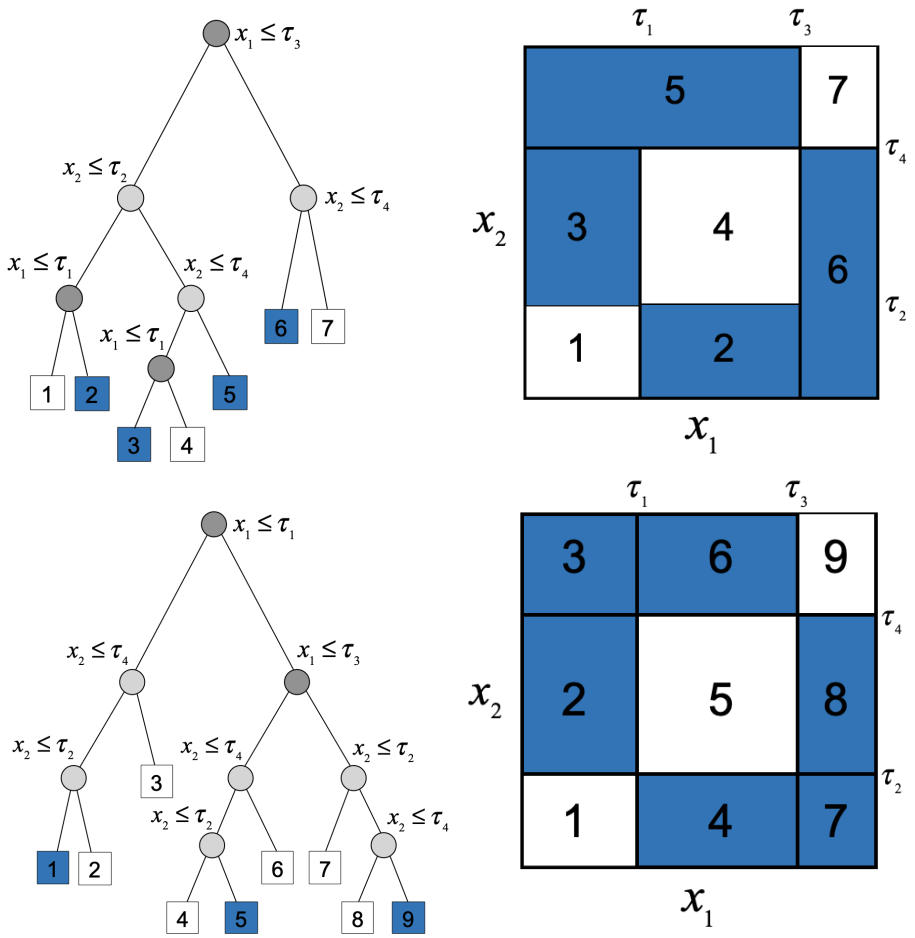


Figure 4.4: Illustration of a tree T_1 (on the top) and a taller equivalent tree T_2 (on the bottom) built using the randomized equivalent tree procedure.

number of distinct splits, the more the splits order in the generated equivalent tree can be "non-optimal" (in term of size) resulting in extremely deep equivalent trees. For that reason we would advice to use this algorithm only on relatively shallow trees to not struggle with computational issues, particularly if policy π is uniform.

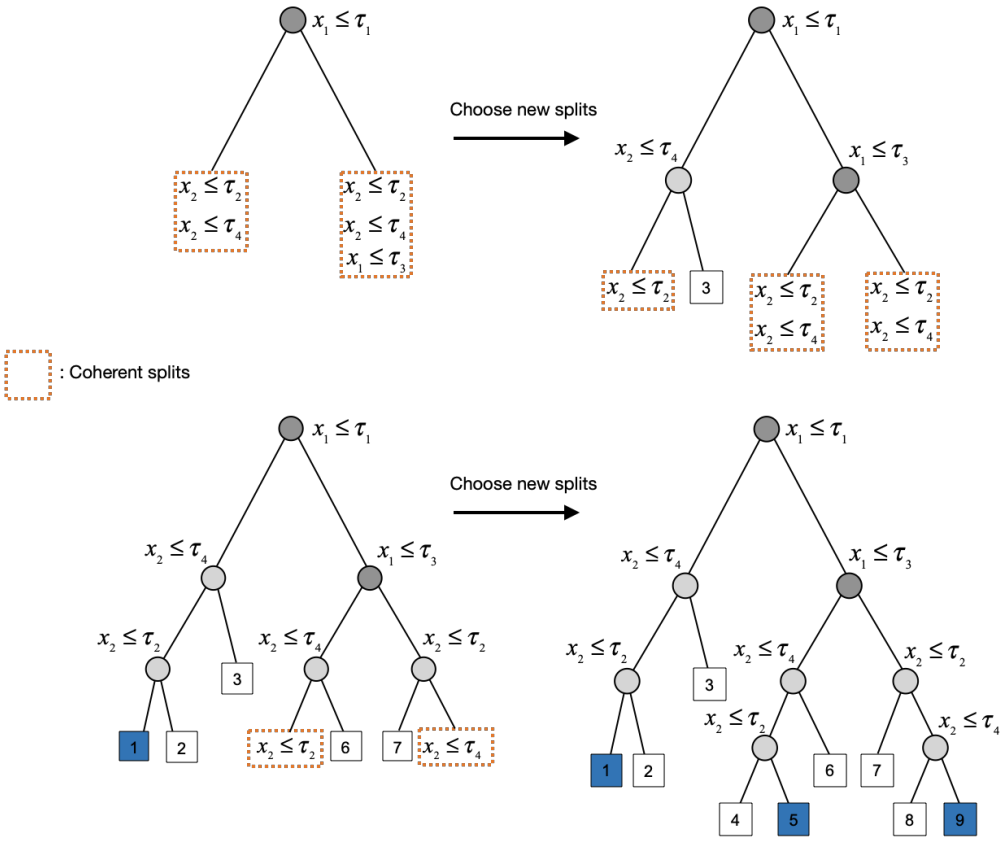


Figure 4.5: Illustration of the recursive randomized equivalent tree generation of T_2 using the same splits as T_1.

Each node of T_2 is chosen randomly from the list of the remaining coherent splits. Once a path of T_2 intersects with a unique label in the partitioning induced by T_1, a leaf of this label is created and the recursive procedure stops.

4 Genetic inspired budget learning using equivalent decision trees

4.1 Genetic algorithms for decision tree optimization

Among global optimization for budget learning in decision trees, genetic algorithms are widely used as non-greedy alternatives to deal with large combinatorial and multi-objective optimizations [17]. They are already largely used in feature subset selection [142, 265], which is very linked to budget learning. For instance, genetic algorithms are introduced to explore the extremely high-dimensional discrete space of all possible feature subsets and retrain successively decision tree models over the selected subset until achieving a fitness or maximum iteration criterion in [14] and [268]. While exploring decision tree models space through genetic algorithms, some inherent drawbacks of local an greedy induction can be avoided [17] as well as keeping the ability of decision tree models to extract compact and relevant set of decision rules [46, 92]. Moreover, as shown in [282] tree structure is well-suited for defining intuitive cross-over and mutation operators and is compatible with string encoding which allows

to use standard genetic programming methods. Most of existing genetic algorithms on decision trees use a standard cross-over consisting in the swapping sub-trees between two decision tree individuals [126]. In this work, as described in Section 4.2, individuals are random forests and the cross-over is based on swapping full decision trees between two random forests. Considering mutation, existing works on genetic algorithm for decision trees propose different mutation operations, depending on the optimization problem they consider, and the correspondent explored space of models.

All of these budget learning methods on decision tree based models aim at minimizing a prediction error while keeping in a limited range of prediction time cost, but they mainly differ on the exact cost function they consider and the way this function is included during the optimization.

4.2 Genetic pruning algorithm for budget learning

This section presents the genetic representation of random forests and the genetic operations used in the algorithm we propose for solving the budgeted prediction time problem. This genetic algorithm deals a population of random forest individuals and their ranking is based on their *fitness value* V_{fit}, assessed using labeled training samples $(x^{(i)}, y^{(i)})$ and defined on a random forest model $\mathcal{M}$ as :

$$V_{fit}(\mathcal{M}) = \frac{1}{n}\sum_{i=1}^{n}\left(\ell(h_{\mathcal{M}}(x^{(i)}), y^{(i)}) + \alpha C(\mathcal{M}, x^{(i)})\right) \quad . \tag{4.13}$$

This comes from the budgeted minimization problem (defined in Section 2.4) over the trade-off between *mean prediction error* and the *mean prediction time cost* of a random forest tuned by the parameter $\alpha > 0$:

$$\hat{\mathcal{M}} = \underset{\mathcal{M}\in\mathcal{H}}{\operatorname{argmin}} \sum_{i=1}^{n}\left(\ell(h_{\mathcal{M}}(x^{(i)}), y^{(i)}) + \alpha C(\mathcal{M}, x^{(i)})\right) \quad . \tag{4.14}$$

The genetic algorithm starts with an already trained random forest and handles this minimization by applying genetic operators depending on this ranking. Random pruning mutation operator ensures to get a lower computation cost but with the risk to increase prediction error, while reproduction operator is applied among the best random forest individuals to mix their genetic information to create new individuals and help reaching better fitness values.

The role of equivalence between decision trees in this optimization is explained and so why it is used the population initialization of the genetic algorithm. As possible splits on which are built random forest individuals is a finite set, the space of random forests explored by our algorithm is also finite, but drastically large. Moreover, optimal equivalent decision trees considering model's size are proved to be NP-hard to find [124, 282] and the considered budgeted learning problem is an extension of this optimal tree search by replacing model size by a more complex computation cost function C an dealing with random forests. Thus genetic algorithms are a common way of dealing with such multi-objective optimization problem [17, 69].

Genetic representation of random forests

Environment	Observations $(x^{(i)}, y^{(i)})$
Individuals	Random forests
Gene	Decision trees
Genome	Trees structure (splits)
Gene regulation	Leaves class values updates
Phenotype	Decision function
Mutation	Genome reduction (pruning)
Reproduction	Tree exchanges between RF

Table 4.3: Analogy between genetic biology and random forest classifiers in our budgeted learning genetic algorithm

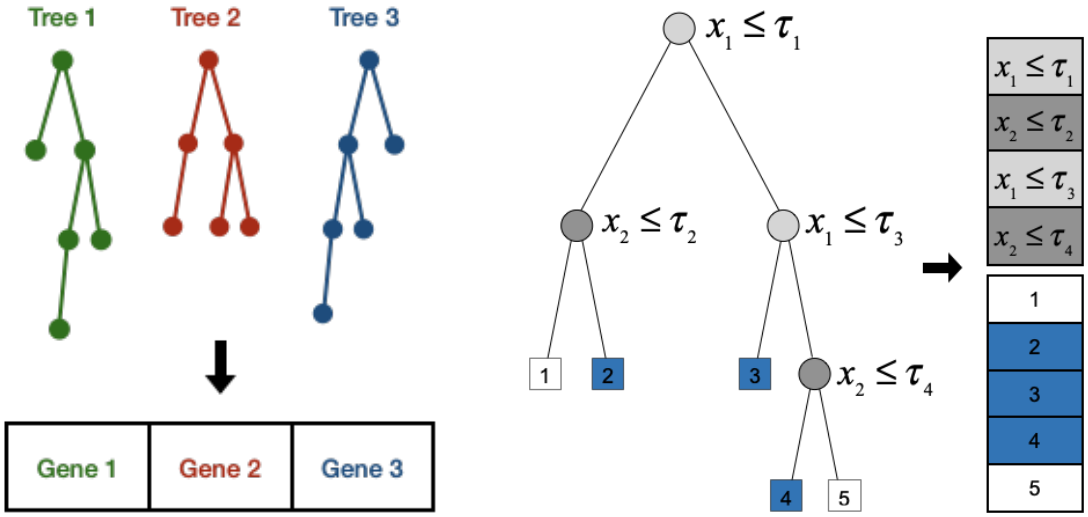


Figure 4.6: Genetic representation of random forests
A random forest genome is a sequence of genes that each represents a decision tree. Then each gene can be encoded into a sequence of splits (feature and threshold) and leaves of the corresponding decision tree.

A common approach for genetic algorithms consists in encoding individuals information into strings that represent individual genomes, then applying general genetic operator defined on strings and finally decoding resulting modified genomes into the original type of individuals.

In this perspective, the genetic representation of random forests we propose is represented in Figure 4.6 and considers each decision tree as a separate gene with two parts which correspond to the ordered sequences of internal nodes and leaves values. Internal nodes are split sequences (feature/threshold couples) and are the part of the genome subject to mutations. These splits are fixed by the initial pre-trained random forest and the genetic algorithm re-orders them in different decision tree structures with randomized equivalent tree procedure. On the other hand, leaf values are label sequences and are updated following the training data of the budgeted learning algorithm, their values being regulated by the environment. For practical implementation simplicity and because decision trees already have suitable structure for genetic operations [17, 126], we do not use this string encoding in this work but it is feasible, as long as cross-over is restricted to certain locations and genome reduction

mutation still keeps strings structure that can be reversely decoded into decision trees.

After the population initialization, each new generation is obtained using the following successive genetic operations : the cross-over reproduction, to increase population diversity and fasten convergence towards a best individual; the pruning based mutation, to explore random forests with lower prediction time costs; and population reduction, to keep a reasonable population size and to increase the mean overall fitness of the whole population. Then a selection and guided mutations are employed to optimize the *fitness value* representing the quality measure of random forest individuals.

Initialization with randomized equivalent trees

Equivalent decision trees are classifiers that differ in their structure although they share the same decision function, Figures 4.3 and 4.4 show examples of equivalent decision trees. In section 2, we explained that the prediction time cost of a random forest is tightly linked to the structure of the trees, as for a given sample x, the cost depends on its paths on the random forest trees. This means that two equivalent decision trees can have different prediction time costs.

Consequently, any budgeted prediction time algorithm applied on a random forest might lead to other solutions while considering an equivalent random forest composed by equivalent decision trees. Our motivation is that exploring the space of equivalent trees from an initial random forest can allow to find other solutions to the budgeted prediction time problem defined in equation 4.11. To illustrate it, we initialize our genetic algorithm with the randomized equivalent trees procedure (pseudo-code 4.1) to give structural variety to the initial population of random forest individuals. This procedure builds, for each tree of the pre-trained random forest, a randomized equivalent tree using the same splits but drawing them randomly in a top-down manner until getting one-class leaves according to the initial tree.

Genome cross-over between random forests

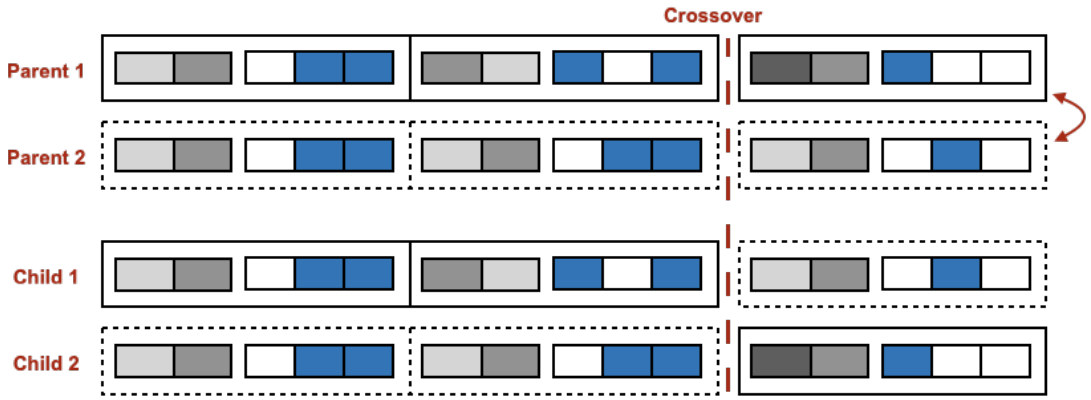


Figure 4.7: Cross-over reproduction between two random forest genomes
Reproducing two random forest parents by exchanging part of their decision trees to create new random forest children is the same as exchanging part of their genes. Then it is comparable to a classic string cross-over operation over random forest genomes, but where random cross-over points are restricted to genes junction points.

The cross-over operation is the usual method of genetic reproduction while using

string encoding. This operation corresponds to the exchange of sub-parts of the parental genomes after at a random location known as the cross-over point. For instance, one common way of doing a cross-over between decision trees is by exchanging sub-trees after having selected cross-over nodes in the two parents.

In this work, the cross-over operation aims to exchange a group of decision trees between two random forest parents. Accordingly to the previously presented random forests genetic representation, it is in other words a restricted locations cross-over that can happen only at genes start/end points, as illustrated in Figure 4.7. Then, over the genetic algorithm iterations, each gene of the random forest individuals has a corresponding ancestor gene corresponding to an initial decision tree or one of its equivalents. Indeed, when dealing with numerical an not quantitative feature like our case, exchanging sub-trees that correspond to non-overlapping regions of the feature space can end up creating unreachable leaves, if it is made completely randomly. Secondly, we want to keep track of the initial random forest model through genetic operations in order to be able, for data analysis purposes, to read the final model as a simplified version of the initial one.

Mutations using random pruning

The second main genetic operation is the mutation and corresponds to the random pruning of random forests individuals. Regarding the prediction time of a random forest, pruning operation ensures a reduction of this cost. According to cost definition in Section 2.4, the prediction time reduction obtained by pruning a decision tree at a given node depends on how much feature usage this pruning avoids and is weighted by the proportion of training samples reaching this node. So each time a random forest individual is selected for mutation a pruning occurs randomly in each of its decision tree, creating a new leaf of the dominant label among training data. To ensure the mutation process to be progressive over iterations, pruning nodes are drawn using an exponential policy relatively to their depth in order to encourage small pruning in terms of deleted sub-trees. As we consider a decision tree as a gene in our representation, this random pruning mutation is a genome reduction mechanism occurring on each gene. Before mutation, concerned individuals are replicated in order to avoid loosing them in the situation where mutation would degrade their fitness value.

Selection and guided mutation

Selection is the way to choose, at every iteration, which individual of the population is targeted by each genetic operation, based on this fitness function ranking. It is inspired by natural selection in biology and relies on the belief that reproducing best individuals while mutating or eliminating the worst ones is likely to result in a best population fitness value improvement over iterations.

In this work, the trade-off function defines the *fitness* of the individuals. Depending on their fitness, organisms undergo a selection step, inspired on the natural selection phenomenon. This operation chooses the individuals with the best fitness (here, lowest $V_{fit}(\mathcal{M})$) to reproduce, and it eliminates the worst individuals (highest $V_{fit}(\mathcal{M})$).

In practice, we used a deterministic selection scheme that picks the β_r ratio of best individuals for reproduction, and the β_d ratio of worst individuals for elimination. Hence, parameters β_r and β_d, correspond to reproduction and death rates respectively. Moreover, we decided to guide the application of pruning mutation operations, using

an exponential policy which is also a function of the fitness value. Then, worst random forest individuals are more likely to be pruned by mutation but creating mutants from best ones is still possible although less likely. This choice is based on the intuition that best individuals should be modified less often in order to preserve them for exploitation, while worst individuals should be modified more often, for exploration purposes.

Algorithm 4.2 Genetic pruning algorithm "BudGenPrune"

procedure $BudGenPrune(\mathcal{R}_f{}^{(0)}, X, Y, N_0, i_M)$
  $pop_0 = RandEqTree(\mathcal{R}_f{}^{(0)}, N_0)$
  for $i = 0 :: i_M$ **do**
    $rep \leftarrow Select(pop_i, \beta_r, X, Y)$
    $pop_i \leftarrow Rep(pop_i, rep, \tau_r)$
    $mut \leftarrow Select(pop_i, \beta_m, X, Y)$
    $pop_i \leftarrow Mut(pop_i, mut)$
    $del \leftarrow Select(pop_i, \beta_d, X, Y)$
    $pop_{i+1} \leftarrow Eliminate(pop_i, del)$
    $\hat{\mathcal{R}}_f \leftarrow$ Minimize $fitness(\mathcal{R}_f, X, Y, \lambda)$
    on pop_{i+1}
  end for
  return $\hat{\mathcal{R}}_f$
  // Best individual of pop_{i_M} according
  to the fitness function.
end procedure

4.3 Experimental setup

This section describes data and costs used to experiment our genetic budgeted learning pruning algorithm, its parameter tuning and the methodology applied to compare and assess the results. Three data sets are used in these experiments : one from synthetic data generation, the *simulated falls* data set and the *real falls* data set. Several definitions of the computation cost function, detailed in section 2, are tested depending on:

- whether the model *evaluation cost* is neglected
- whether the *feature acquisition cost* is considered data dependent
- whether feature groups costs (corresponding to signal representation computations) are included

4.3.1 Data

Synthetic data used in experiments are generated from binary labeled gaussian clusters with the synthetic generator presented in the previous chapter for transfer learning on synthetic data. The gaussian distributions are composed by 10 clusters of each label with mean μ and variance σ^2 parameters drawn randomly between bounded values (respectively $\mu \in [-50, 50]$ and $\sigma^2 \in [5, 15]$). Samples are generated with $d = 20$ features with 5 to 10 of them that are randomly chosen to be white noise and the rest being

informative. As there is no real feature acquisition cost for synthetic data we choose to set their values with their estimated *feature importance* relatively to the original random forest to emphasize the trade-off aspect between their discriminant efficiency and their costs. Usually the time spent k_{ev} to compute one node split is significantly lower to any feature cost, but for synthetic data no comparison between these two types of costs can be done as they have to be arbitrarily set. So for these synthetic data we consider only *feature acquisition cost* and not *evaluation cost*, meaning that only used features and their order matter and not the model size.

Tarkett fall detection data are also used in the following experiments and come both from simulated and real falls and compose two different databases presented in section 4 of the first chapter. As the first implementation of the fall detection model is known to fit embedded system prediction time budget it is used as a comparison reference. The initial random forest classifier is trained similarly to the reference model using one part of simulated data and then two main scenarios are presented : for data independent cost functions (when prediction time is considered to be the same for every observation) another part of simulated data is used to train the budgeted learning genetic algorithm whereas for data dependent cost functions we used real data to train the algorithm. This is motivated by the fact that the mean prediction time we try to estimate should be more representative to real conditions using real falls database (for instance fall events proportion is closer to reality) and it is also a way to test the behavior of the algorithm while confronted with a prediction time cost assessment domain that differs from the initial training domain. Every signal representation and feature described in Chapter 2 is used in these experiments and their costs have been measured empirically in mean by repeating the specific corresponding computations.

4.3.2 Classification models and budgeted learning algorithm

Experiments are done using random forests of 10 binary classification decision trees, with a maximum depth set to 7 (allowing up to $2^7 = 128$ leaves for each tree). Each dataset is equally separated into 3 subsets : the initial training set to train the original random forest model using CART algorithm, a training set for the genetic pruning algorithm to assess the *fitness value* of every model at each iteration and a last subset to test the value of the final budgeted random forest model selected by the algorithm.

Every random forest assessed by the genetic pruning algorithm uses uniquely splits that are already present in the initially trained one, like the budgeted pruning algorithm presented in [184]. Nevertheless our method allows different order of these splits in decision tree paths, while using randomized equivalent trees generation for initialization. This makes the space of random forest candidates a lot wider than in the compared work[184] optimal pruning framework and this is the reason why in our work we prefer a genetic programming approach rather than a combinatorial optimization.

One particularity of our genetic algorithm is that it works the same way independently to the cost function definition. This allows to test it on our fall data 4 different ways of defining the C cost function depending on whether it considers the same model cost for every sample or not and whether feature groups framework is considered (in our case these groups are signal representations). Details about these different definitions are explained in Section 2.3.

To observe whether using equivalent decision trees can be relevant or not for our budgeted learning purpose, each experiment is done on two separate population starting from the same initial random forest. The first population is obtained with equivalent trees, as described in section 4.2, whereas the second one is obtained by exact duplication of the initial random forest. For each experiment, datasets are split into three subsets : the initial random forest training set, the training set used by genetic algorithm and a test set to assess final best random forest individual after all iterations. The main performance measure considered is the trade-off between mean accuracy and mean prediction time cost represented by the fitness function.

Each experiment is repeated 10 times and the best random forest individual is tracked over iterations of the genetic algorithm. Population size at the start of the genetic algorithm is 40 random forests obtained from the initially pre-trained model (either by randomized equivalent trees procedure or duplication). At each iteration, new individuals are added to the population through mutation and reproduction whereas the less valuable ones according to the fitness function are removed. Mutation rate and reproduction rate are respectively set to $\beta_m = 0.7$ and $\beta_r = 0.15$. Birth rate and mortality rate are set to $\tau_r = 3$ and $\beta_d = 0.5$. Thus, the population size variation from a generation to the next one is given by the factor : $(1 + \beta_m)(1 + \beta_r * \tau_r)(1 - \beta_d)$ which is around 1.23 in our experiments, meaning the population is slowly growing over iterations. All the considered feature costs are normalized such as the sum of all possible costs is equal to 1 and the accuracy/prediction time trade-off parameter is set to $\alpha = 1$. Python implementation of algorithms 4.1 and 4.2 are available online [2, 3].

4.4 Results

This section presents the results obtained with the previously mentioned experiments, and our interpretations. Figures 4.8 and 4.9 illustrate the algorithm's behavior on the best random forest individual during iterations on datasets used for the budgeted training whereas Table 4.4 shows final best individual results assessed on the test datasets (to simplify the display of these results, we refer to the prediction time cost as "Budget").

Obviously, due to the feature space high dimension, for experiments on fall detection data all the features are not present in the initial random forest, which explains that the prediction time cost starts from a value lower than 1 contrary to synthetic data experiments. Figures 4.8 and 4.9 illustrate that, using parameter values given in previous section, the genetic algorithm seems to converge on all our experiments after a few iterations towards a best random forest individual regarding the fitness function value. This shows firstly that using pruning mutation in combination with tree exchanges based cross-over is a valid method to optimize the trade-off between accuracy and prediction time cost of a random forest. As we can observe both on synthetic data and fall detection data, the fitness value decrease is very linked to the prediction time cost reduction (between 5% and 10% for the synthetic data and between 40% and 70% for fall detection data), which is the main purpose of the algorithm. Depending on the inherent complexity of data distribution with regards decision trees partitioning and the value of α, this prediction time reduction can lead to a variable loss in accuracy (as represented in Figure 4.8), but fall detection models almost keep

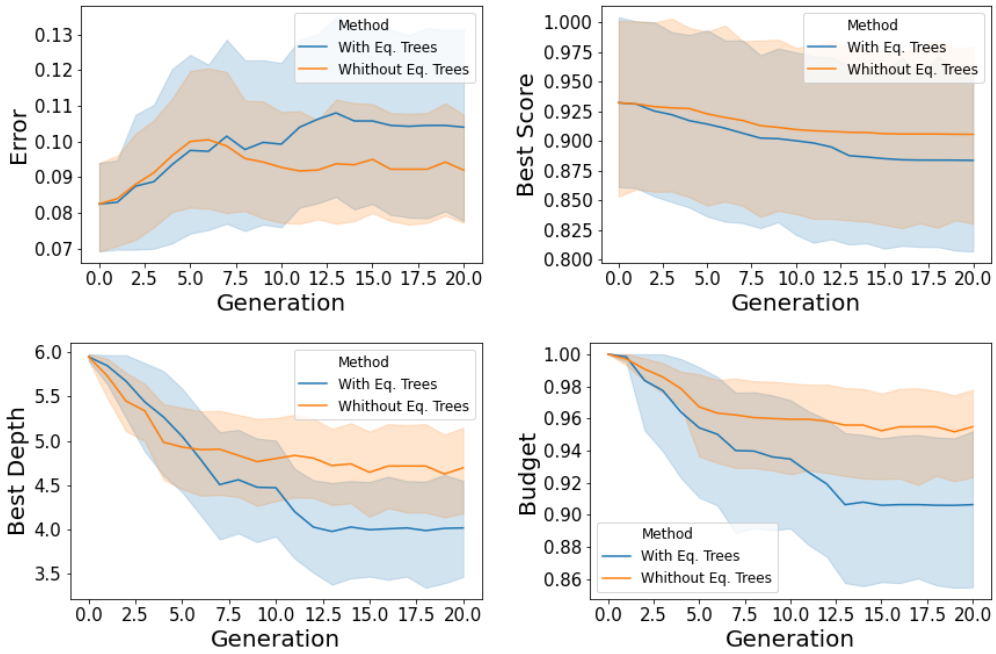


Figure 4.8: Error, fitness value, budget and depth of the best RF over BudGenPrune algorithm iterations and the impact of equivalent decision trees initialization on synthetic data.

the same prediction accuracy while the prediction time is importantly reduced (as represented in Figure 4.9).

	Data independent			
	No group costs		Group costs	
	With Eq.	Without Eq.	With Eq.	Without Eq.
Fitness value	0.14 ± 0.03	0.18 ± 0.05	0.29 ± 0.07	0.37 ± 0.07
Budget	0.091 ± 0.02	0.13 ± 0.03	0.22 ± 0.05	0.30 ± 0.03
Error	0.068 ± 0.01	0.073 ± 0.01	0.075 ± 0.01	0.084 ± 0.01
Depth	2.8 ± 0.3	2.3 ± 0.2	2.5 ± 0.4	2.1 ± 0.1
N° nodes	46	34	54	36
N° features	16	12	29	21

Table 4.4: Budget-wise comparison of different random forest models for fall detection embedding considering a data-independent prediction time.

Another aspect of these results is the impact of using equivalent decision trees. Indeed figures show that the genetic algorithm seems to reach better values of *fitness function* and prediction time with the population evolved from randomized equivalent trees initialization. This phenomenon is as pronounced on fall detection data as on synthetic data, and this difference in prediction time reduction using equivalent trees is especially interesting insofar as it does not necessarily reflect an accuracy difference as shown in Figures 4.9 (c) and 4.9 (d) . Overall, all these figures reveal that using equivalent trees with this genetic algorithm improves the budget learning task on our

data but it tends to take more iterations to converge. Also the mean error and prediction time variance of the best random forest individual over several repetitions appears to be a bit higher using equivalent trees. These observations are consistent with the intuition that space of possible prediction time/accuracy couples of reachable solutions is highly wider with equivalent trees use because of the structural variety it induces. It is moreover confirmed by the fact that using equivalent trees, the algorithm can in some cases reach better *fitness value* and lower *prediction time* although converging towards deeper trees compared to not using equivalent trees (Figures 4.10 (a),(b),(c)).

Furthermore it is interesting to remark the impact of the different definition of *prediction time cost* function. In *data dependent* situation prediction time costs are weighted by the amount of samples reaching each node, which allows to obtained deeper trees on the budgeted random forest, as shown in Figures 4.10. Taking into account feature group costs in C definition provokes an interesting "drops" behavior of the total budget and fitness value at some point during iterations (Figures 4.9(b),(d) and 4.10(b),(d)). This corresponds to a complete signal representation cost being amputated after pruning every feature of this group and this is only observable using equivalent trees as it needs features reordering making possible for all features of one group to be placed at the bottom of trees.

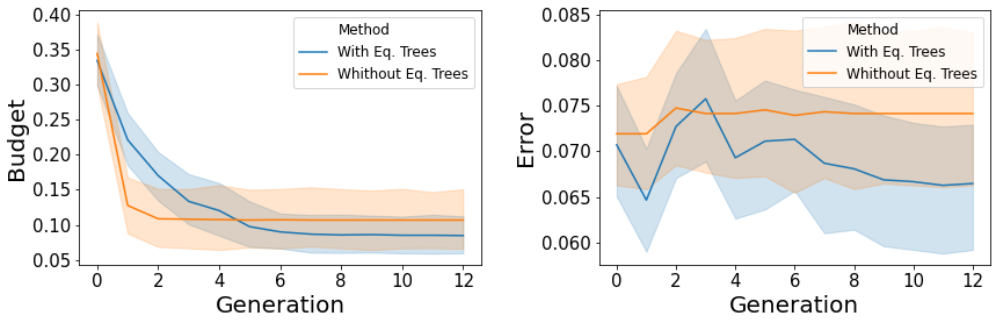
	Data dependent			
	No group costs		Group costs	
	With Eq.	Without Eq.	With Eq.	Without Eq.
Fitness value	0.16 ± 0.04	0.20 ± 0.03	0.32 ± 0.06	0.36 ± 0.03
Budget	0.052 ± 0.01	0.081 ± 0.04	0.26 ± 0.08	0.31 ± 0.02
Error	0.12 ± 0.05	0.14 ± 0.03	0.089 ± 0.01	0.072 ± 0.03
Depth	2.2 ± 0.2	2.5 ± 0.5	2.7 ± 0.4	3.2 ± 0.8
N° nodes	40	36	36	46
N° features	21	18	20	23

Table 4.5: Budget-wise comparison of different random forest models for fall detection embedding considering a data-dependent prediction time.

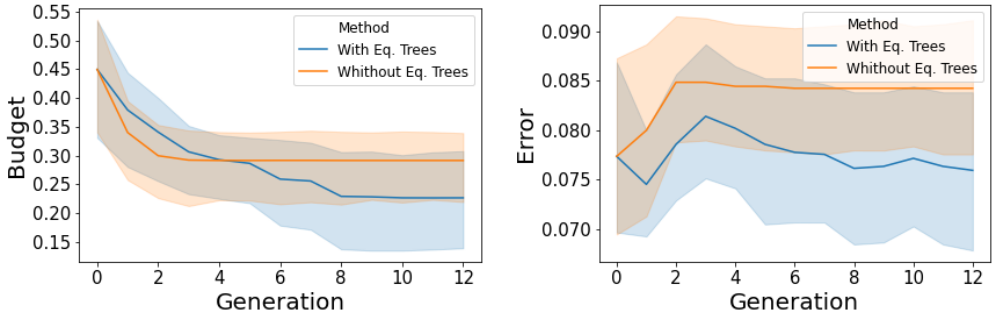
Finally a surprising phenomenon is noticeable on fall detection data experiments with data dependent prediction time cost definitions (Figures 4.9 (c) and 4.9 (d)). Indeed, as previously explained the initial random forest is trained on the simulated fall database whereas real fall database is used to assess error and prediction time cost during BudGenPrune algorithm iterations. On these experiments we can observe that not only the mean prediction time is decreasing but also the mean error at the same time. This means that pruning mutations also serves to increase prediction accuracy from the *source* model on *target* data, which could be considered as a form of transfer learning. More importantly it shows that using equivalent trees could also have utility before applying model-based transfer algorithms like SER and STRUT, that are dependent to the structure of decision trees they are applied on.

Concerning our embedded fall detection application, this genetic pruning algorithm allows a non-negligible reduction of the prediction time without randomized equivalent trees and even more while using them. From the 128 initial features, final best random forest individuals keep in mean around 20 to 40 features representing in most cases less than 15 % of all features costs, and are also a lot lighter in terms of depth and

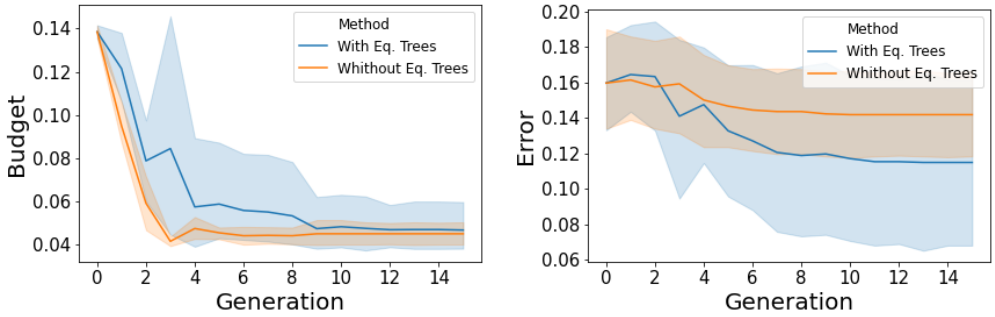
number of nodes. To compare Table 4.4 presents results obtained in these experiments in terms of accuracy and several prediction time aspects in front of the first version of random forest implemented in 2017 in our embedded systems for fall detection [174]. The proposed genetic pruning approach with equivalent trees directly considers the mean prediction time budgeted by a given device to simultaneously reduce the *feature acquisition* cost (through features reordering and feature usage lowering) and the *evaluation cost* (through depth decrease).



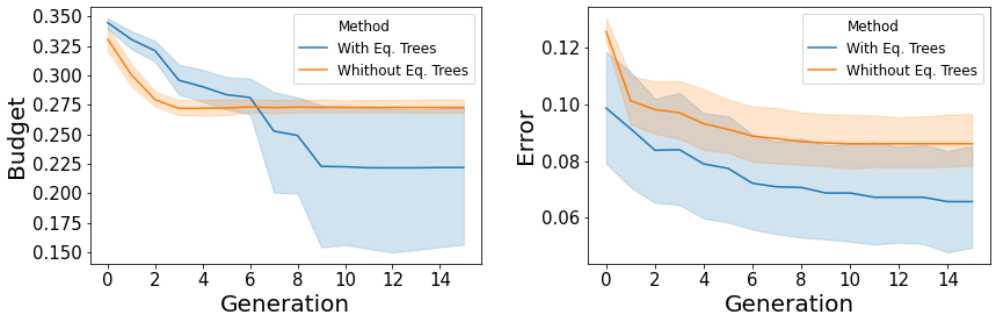
(a) Data independent prediction time without feature group costs



(b) Data independent prediction time with feature group costs

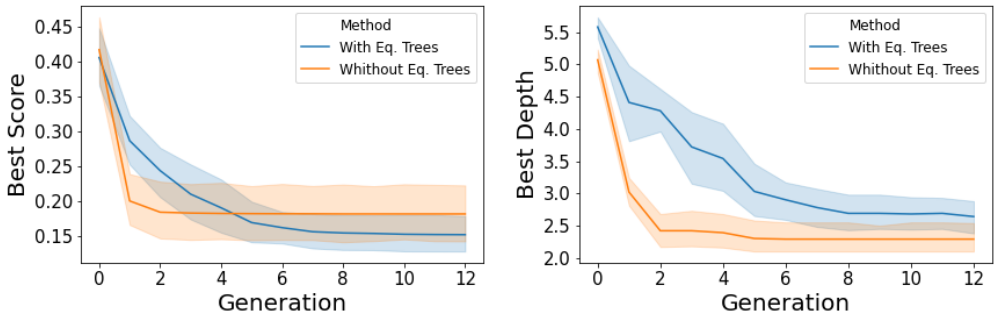


(c) Data dependent prediction time without feature group costs

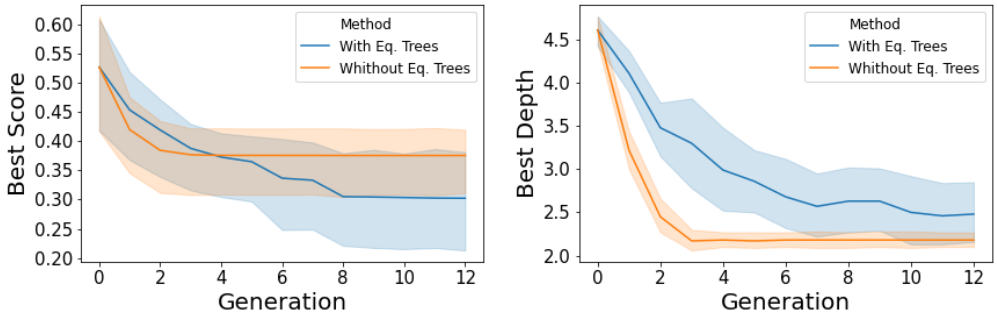


(d) Data dependent prediction time with feature group costs

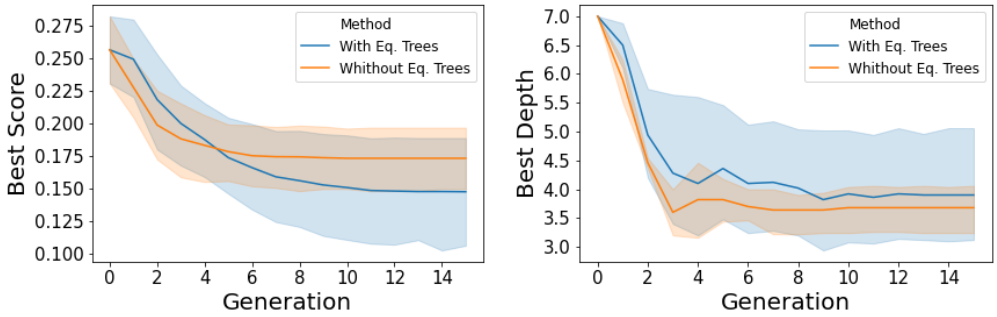
Figure 4.9: Mean error and prediction time of the best RF over BudGenPrune algorithm iterations and the impact of equivalent decision trees initialization on fall detection data.



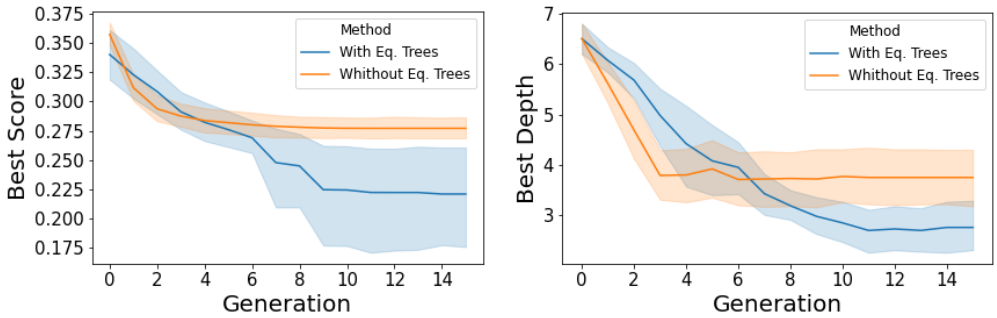
(a) Data independent prediction time without feature group costs



(b) Data independent prediction time with feature group costs



(c) Data dependent prediction time without feature group costs



(d) Data dependent prediction time with feature group costs

Figure 4.10: Mean fitness value and tree depth of the best RF over BudGenPrune algorithm iterations and the impact of equivalent decision trees initialization on fall detection data.

5 Conclusion and perspectives

The field of budgeted learning is presented in this chapter as well as why it is relevant for practical applications. In particular machine learning applications that relies on real-time prediction and/or embedded systems are especially concerned with resource constraints that can be included in the learning process. Thus Tarkett elderly monitoring application is a typical example of the need of budgeted learning and this work focuses on constrained prediction time of random forests, experimented on the fall detection task. For this purpose we formalize a general definition of computation costs on decision tree based models and precise it in our specific context. In order to solve the budgeted prediction time problem we propose an innovative genetic algorithm based on random pruning and equivalent decision trees that shows its efficiency in reducing importantly the random forests prediction time while keeping the prediction accuracy almost at the same level.

5.1 Computation costs

This chapter opens the path to a first kind of perspectives concerning the variety of computation costs. As previously mentioned working on an embedded system implies several constraints related to its volatile memory (RAM), non-volatile memory (ROM) and computing power. As for our application critical resource constraints were neither ROM or RAM related, our work is focused on the model prediction time which mainly depends on the computing power. Nevertheless nowadays the size of machine learning models can be so massive that their impact on non-volatile memory is no longer negligible [240]. Moreover if the ROM part allocated to store a model is easily deducible from its size, the RAM consumption is more complex to formalize. Indeed it varies in time according to the order of computations and involves every temporary computation, which can represent an important part of the RAM (e.g. spectrograms). Thus one extension of this work would be to integrate all these kinds of computational constraints into a united budgeted learning framework for embedded systems.

Concerning temporary computations we show through our example of signal representation computation costs that several features can share some part of their computations, making their costs inter-dependent. Even if this only one simple degree of dependency integrating these additional *feature group costs* impacts importantly the total prediction time cost and therefore the whole budgeted learning optimization. In practice links between feature computations can be of various forms and far more intricate. Then by considering these links, a graph of dependency between all the computations can be built and optimizing computations accordingly for a fixed set of features becomes an interesting problem in itself. Moreover another level of dependency can involved by including possible recursive computations while features are computed in real-time over time series.

Finally as mentioned in this chapter introduction resource constraints and budgeted learning strategies does not concern only final models and prediction phase. For Tarkett monitoring application, although data gathering and storage capabilities are important, they are still limited in terms of the total volume and the amount of data that can be collected at the same time and they imply financial costs. Moreover labeling new data is still a sensitive task and remained manual throughout this project. Thus

developing active strategies of data selection and labeling, with regards to these server side computational limits could be a major long-term improvement. Such problems are at the cross-point between online learning, active learning and budgeted learning fields [104, 176, 280] and concern numerous real applications [163, 253].

5.2 Equivalence of classification models

Another key point this chapter treats is that some performance measures, like prediction time cost, do not depend only on the decision function of a classification model but also its internal structure. We illustrate it by distinguishing a classification model $\mathcal{M}$ and its decision function $h_{\mathcal{M}}$ and by developing the notion of *equivalence* between classification models. This distinction is rarely done in classic machine learning for the following reason. As long as metrics used to assess classification models depends uniquely on the decision function, like usual *loss functions*, the notion of equivalence has no direct impact. Nevertheless, even with these kinds of metrics, this notion can be impacting from the moment some structural modifications are applied on a pre-existing model.

Let's assume an already trained classification model $\mathcal{M}$, an equivalent model $\mathcal{M}_{eq}$ and n training data $S = \left((x^{(1)},y^{(1)}),...,(x^{(n)},y^{(n)})\right)$ are available. Denote $\mathcal{R}$ a performance measure, like the mean prediction error, that depends only on decision function. Then $\mathcal{M}$ and $\mathcal{M}_{eq}$ have of course the same performance $\mathcal{R}(h_{\mathcal{M}}) = \mathcal{R}(h_{\mathcal{M}_{eq}})$.

Now suppose a deterministic algorithm $\mathcal{A}$ is applied on $\mathcal{M}$ using data set S, producing a new model $\mathcal{A}_S(\mathcal{M})$. If $\mathcal{A}$ depends on the structure of the model $\mathcal{M}$, like $SER, STRUT$ or any pruning algorithm for example, then it can result in different models $\mathcal{A}_S(\mathcal{M}) \neq \mathcal{A}_S(\mathcal{M}_{eq})$ of different decision function $h_{\mathcal{A}_S(\mathcal{M})} \neq h_{\mathcal{A}_S(\mathcal{M}_{eq})}$, and therefore may entail this interesting situation :

$$\exists \mathcal{M}_{eq} \sim \mathcal{M} : \mathcal{R}(h_{\mathcal{A}_S(\mathcal{M})}) \neq \mathcal{R}(h_{\mathcal{A}_S(\mathcal{M}_{eq})})$$

In those cases, questioning the impact of the equivalence class of a model $\mathcal{M}$ on the algorithm $\mathcal{A}$ could be useful to consider. While dealing with such kind of machine learning problem, this idea of "structural dependency" can have an influence through two main aspects.

A first degree of "structural dependency" relies on the measure $\mathcal{R}$ and whether it depends on the structure of the model $\mathcal{M}$ or only on the decision function $h_{\mathcal{M}}$. For instance measures that take into account the complexity of the model, which is typical in budgeted learning, are structure dependent. That is why there is an advantage of using equivalent decision trees for our budgeted prediction time problem. Moreover a second degree of dependency corresponds to the algorithm $\mathcal{A}$ itself. Indeed if replacing a model $\mathcal{M}$ by an equivalent one $\mathcal{M}_{eq}$ has no impact on the output classifier decision function after applying the algorithm $\mathcal{A}$, then it can be qualified as "structure independent" and therefore can be defined directly on decision functions.

Definition 4.4. *Structural dependency*
Using notations previously defined if $\forall(\mathcal{M},\mathcal{M}_{eq}) \in \mathcal{H}, \forall S \in (\mathcal{X} \times \mathcal{Y})^n$,

$$\mathcal{M}_{eq} \sim \mathcal{M} \implies h_{\mathcal{A}_S(\mathcal{M})} = h_{\mathcal{A}_S(\mathcal{M}_{eq})} \tag{4.15}$$

Then algorithm $\mathcal{A}$ *is* ***structure independent*** *and its action can be defined directly on decision functions :* $h_{\mathcal{A}_S(\mathcal{M})} = \mathcal{A}_S(h_{\mathcal{M}})$

For instance algorithms achieving label permutation or uniform translation along an axis are directly defined on the decision function and are *structure independent*. On the contrary, in order to show that an algorithm is not *structure independent*, pointing out an example of two equivalent models such that equation 4.15 is not satisfied is sufficient.

However definition 4.4 is not enough to differentiate between two structure dependent algorithms and does not permit to characterize the amplitude of possible changes while applying an algorithm $\mathcal{A}$ on equivalent classification models. Thus finding a good way to quantify this *structural dependency*, maybe with regards to the measure $\mathcal{R}$, would be an interesting perspective that could lead to various relevant uses of the equivalence notion. To this end it is conceivable to slightly different definition of *structural dependency* and a less strict versions of the equivalence definition between classification models.

Taking advantage of equivalent models can be possible from the moment a structure dependent algorithm is applied on a pre-trained model. One direct use we can think of concerns heterogeneous transfer learning, while some features that are present in the source domain are not available on target domain. As it is possible to force some features to be at the extremities while generating equivalent decision trees, these features are more easy to remove from the model by pruning before applying an usual transfer algorithm like SER or $STRUT$.

Furthermore equivalence in terms of decision function between classification models is not specific to decision trees and can be defined in the same way on other families of classifiers like neural networks. For example authors in [25] present a method to create a neural network, namely *neural forest*, that is equivalent to a given random forest in order to initialize more efficiently the architecture of the model before a training phase. Finally even without considering the performance of classification models, equivalence could be used to help interpretation. For instance for any random forest classifier, there exist strictly equivalent decision trees (and then an optimal one regarding the size), that might be precious for interpretation purposes. Although this perspective is particularly attractive for practical applications, as explained in Section 3, efficient heuristics to approach optimality are hard to find.

5.3 Genetic algorithm for budgeted learning

The genetic algorithm on random forests developed in this chapter shows its efficiency regarding the budgeted prediction time problem but numerous areas of improvement are still remaining considering all the various aspects it involves. The first one concerns random policies used by the algorithm that act at three main levels : the reproduction when choosing from parent random forests the trees that will constitute children, the mutation where pruned nodes are drawn depending on their depth following an exponential policy, and during the randomized equivalent tree procedure which for each new node selects randomly a split with an uniform policy from the list of consistent splits. These policies do not integrate any computation cost information so intuitively it could be more efficient to take into account feature costs during the randomized equivalent tree generation or the total prediction time cost gain while drawing nodes to prune. Nevertheless these kinds of strategies can be risky because

even if it can speed up and facilitate the convergence toward a solution it also reduces the amplitude of exploration of potential solutions.

Moreover, concerning genetic operations, we use the randomized pruning as the only type of mutation, and equivalent trees are only intervening in the algorithm initialization. Other kinds of mutations can help to reduce the prediction time cost like replacing some features by others or removing one tree from the forest for example. Indeed this algorithm while converging usually reduces the mean depth of the initial random forest but the total number of decision trees stays unchanged. So removing one tree would be a form of bigger mutation than pruning. Also while decision trees are not very deep, creating a new tree equivalent to a small part of the random forest (the complexity of equivalent generation procedure depends on the total amount of considered splits) is conceivable as another way of reducing random forest size and could serve as a new form of reproduction or mutation.

Concerning the different versions of the budgeted prediction time problem, although the presentation of the different definitions of random forests prediction time and the budgeted problem provided in Section 2 is not exhaustive, it shows the wide variety of situations. As our algorithm is very flexible it can easily be extended in order to deal with several constraints at the same time by adapting the fitness function. For instance it could handle budgets both on the mean prediction time and worst case prediction time as well as a simultaneous constraint on the total model size in memory. Finally a budgeted learning framework that has not been taken into account in our experiments concerns early stopping models that represent an interesting perspective for the budgeted prediction time problem on random forests. It seems to be at the same time particularly adapted to decision trees and a less restrictive alternative compared to more hardly constrained models.

Conclusion and perspectives

This thesis relates to an elderly activity monitoring project of Tarkett, namely FIM Care, based on a piezo-electric flooring sensor and using supervised decision tree based models in an embedded system. It provides several contributions in different topics related to the industrial application of machine learning models that arise from real implementation problems encountered throughout the development of this project. As a starting point we present the state of literature concerning human activity monitoring system in terms of sensors, monitoring tasks, feature extraction and type of model used in these applications. According to these aspects, we describe Tarkett technology specificity and compare it with these existing human activity monitoring systems. Using data obtained in various experimental environments and specific predictive models based on random forests, we explore the feasibility of several monitoring classification tasks such as *walk recognition*, *lying on-floor* detection and *fall detection*, the latter having been implemented in real conditions in 2017. Concurrently we provide some insights about time series representation, feature extraction and their redundancy as well as their relevance relatively to each detection tasks.

Using first implementations of the fall detection algorithm on real environments during more than one year, data have been gathered into a new database composed by real falls and non-fall events. Motivated by the observation of dissimilarities between experimental and real datasets, we explore the problem of model-based *transfer learning*. As falls occur far more rarely than other events and are more complex to collect, we focus our work on the impact of *class imbalance* and relative data rarity in target domain while attempting to transfer random forests trained on a source domain where classes are equally represented. Based on the *pruning risk* notion, we propose adaptations of two seminal transfer algorithms on decision trees to address this *class imbalance* problem. We illustrate the benefits of our variants on several kinds of imbalanced data and combine them into a general selective transfer algorithm on random forest (STRF) able to cope with various domain shift situations. Our results suggest furthermore that the selection process of this algorithm could help to characterize the need and the nature of the transfer.

Finally for the concrete FIM Care monitoring application any of the embedded predictive model has to be constrained in term of the computational resources it requires. This issue is presented as a *budgeted learning* problem where the prediction time of a random forest model is constrained. This prediction time is modeled, in accordance with the literature, by the sum of two costs : the *feature acquisition* cost and the *evaluation* cost. We propose to tackle the constrained prediction time problem with a genetic algorithm using random pruning mutations and taking advantage of *equivalent decision trees* that capitalize on feature reordering inside trees structure to enhance *feature acquisition* cost reduction. Our approach manages to successfully reduce the prediction time of our tested random forest models in exchange for very few loss in prediction accuracy. Another key aspect of this method is that the obtained budgeted model is still a pruned version of an equivalent random forest of the initial

model. This is quite valuable for our monitoring application, as we can keep an original unconstrained random forest offline for detailed interpretation, while a constrained version is embedded for efficient real-time online predictions.

Numerous perspectives come out of our contributions on several aspects of this work. Firstly our results on fall detection illustrate that smoothing random forest instantaneous predictions helps to reduce the false alarm rate, which suggests that the series of decision tree outputs might contain temporal information that could be further exploited. Indeed in the presented predictive models temporal information is encapsulated in some of the features but considering instantaneous responses of decision trees as different states of a Markovian model could be an interesting continuation that might lead to some improvement in classification accuracy. Apart from classification performance enhancement our approaches on *transfer learning* and *budgeted learning* can also be extended to other application contexts. Indeed we worked on transfer algorithms that are focused on *homogeneous transfer*, dealing with the same tasks and feature space from source domain to target domain. So one part of our future work will be to adapt these algorithms to *heterogeneous transfer* situations, where different but related tasks like *lying on-floor* detection and *fall detection* could be transferred from one to the other. Considering *budgeted learning*, we introduced the notion of inter-dependency between feature computations through the simple case of shared acquisition costs of *grouped features* corresponding to signal representation computations. This idea can be push further with more complex relations between features computations as it is often the case in practice while dealing with a large pool of features computed from the same original signal. Finally the structural exploration offered by the notion of *equivalence* of classifiers can be defined on other kinds of model relying on graph structures (like neural networks) and might be interesting for other purposes than *budgeted learning*. One direct use case would be for example to obtain a more interpretable model by translating efficiently a classification random forest into an equivalent decision tree classifier.

A

Features definition

In the following we consider $\mathbf{s} = (s_1,...,s_N)$ a discrete signal portion of size N, $\mathbf{a} = (a_1,...,a_N)$ the corresponding autocorrelation vector, $(f_1,...,f_{\frac{N}{2}})$ the coefficients of its fast Fourier transform (and $\mathbf{f} = (|f_1|,...,|f_{\frac{N}{2}}|)$ the vector of their modules) and the matrix W of size $(N, \frac{N}{2})$ of the modules of Wigner-Ville spectrogram coefficients.

We indicate by $b^+_{peak}(s_i) = 1$ if s_i is a positive "peak" of $\mathbf{s}$ and $b^-_{peak}(s_i) = 1$ if it is a negative "peak" such that :

$$b^+_{peak}(s_i) = \mathbb{1}\{s_i > v / s_i - s_{i-1} > \Delta, s_i - s_{i+1} > \Delta\},$$
$$b^-_{peak}(s_i) = \mathbb{1}\{s_i < -v / s_i - s_{i-1} < -\Delta, s_i - s_{i+1} < -\Delta\},$$

with v and Δ positive parameters. In the same way we denote the according values of these peaks :

$$v^+_{peak}(\mathbf{s}) = \{s_i, s_i > v / s_i - s_{i-1} > \Delta, s_i - s_{i+1} > \Delta\},$$
$$v^-_{peak}(\mathbf{s}) = \{1 < i < N-1 / s_i < -v, s_i - s_{i-1} < -\Delta, s_i - s_{i+1} < -\Delta\},$$

and their corresponding indexes :

$$i^+_{peak}(\mathbf{s}) = \{s_i, s_i > v / s_i - s_{i-1} > \Delta, s_i - s_{i+1} > \Delta\},$$
$$i^-_{peak}(\mathbf{s}) = \{1 < i < N-1 / s_i < -v, s_i - s_{i-1} < -\Delta, s_i - s_{i+1} < -\Delta\}.$$

Then vectors of gaps between these peaks are :

$$\Delta^+_{peak}(\mathbf{s}) = i^+_{peak}(\mathbf{s})[2 :: n^+_{peak}(\mathbf{s})] - i^+_{peak}(\mathbf{s})[1 :: n^+_{peak}(\mathbf{s}) - 1],$$
$$\Delta^-_{peak}(\mathbf{s}) = i^-_{peak}(\mathbf{s})[2 :: n^-_{peak}(\mathbf{s})] - i^-_{peak}(\mathbf{s})[1 :: n^-_{peak}(\mathbf{s}) - 1].$$

Finally the vector of durations of signal parts where its absolute value is below a threshold v is denoted :

$$\Delta_t(\mathbf{s}) = \{j - i \quad / \quad 0 < i < j < N+1, \forall k \in [i,j], |s_k| < v, |s_i| > v, |s_j| > v\}.$$

i_m and i_M : indexes of max. and min. of $\mathbf{s}$. (t_{max}, f_{max}) : indexes of max. of W module and (t_{bary}, f_{bary}) are coordinates of barycenter of W module.

Feature name	Parameter	Definition	Signal representations
Maximum		$\max\{s_1,...,s_N\}$	Raw signal, Derivative, Integral
Minimum		$\min\{s_1,...,s_N\}$	
Median		Median of $\{s_1,...,s_N\}$	
Absolute median		Median of $\{\|s_1\|,...,\|s_N\|\}$	
Mean		$\mu = \frac{1}{N}\sum_{i=1}^{N} s_i$	
Absolute mean		$\frac{1}{N}\sum_{i=1}^{N} \|s_i\|$	
Standard deviation		$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(s_i-\mu)^2}$	
k-th moment	k	$M_k(\mathbf{s}) = \frac{1}{N}\sum_{i=1}^{N}\frac{(s_i-\mu)^k}{\sigma^k}$	
Energy		$\frac{1}{N}\sum_{i=1}^{N} s_i^2$	
Log-energy		$\frac{1}{N}\sum_{i=1}^{N} log(1+s_i^2)$	
Shannon energy		$\frac{1}{N}\sum_{i=1}^{N} s_i^2 log(1+s_i^2)$	
Delta min-max		$max - min$	
Nb-sup-threshold	v	$\sum_{i=1}^{N} \mathbb{1}\{s_i > v\}$	
Peak count	v,Δ	$n^+_{peak}(\mathbf{s}) = \sum_{i=2}^{N-1} b^+_{peak}(s_i)$	
Neg-peak count	v,Δ	$n^-_{peak}(\mathbf{s}) = \sum_{i=2}^{N-1} b^-_{peak}(s_i)$	
Delta-before-max		$s_{i_M} - s_{i_M-1}$	
Delta-after-max		$s_{i_M} - s_{i_M+1}$	
Delta-before-min		$s_{i_m} - s_{i_m-1}$	
Delta-after-min		$s_{i_m} - s_{i_m+1}$	
Inf-threshold-ratio	v	$\frac{1}{N}\sum_{i=1}^{N} \mathbb{1}\{s_i < v\}$	
Inf-threshold-duration	v	$\mu(\Delta_t(\mathbf{s}))$	
Percentile	α	$perc(\mathbf{s},\alpha)$	
Inter-percentile	α_1,α_2	$perc(\mathbf{s},\alpha_2) - perc(\mathbf{s},\alpha_1)$	
Log-mean-peak		$log(1+\mu^+_{peak}(\mathbf{s}))$	
Log-mean-neg-peak		$log(1+\mu^-_{peak}(\mathbf{s}))$	
Log-mean-peaks-diff		Log-mean-peak - Log-mean-neg-peak	

Table A.1: Features extracted from raw signal, derivative and integral.

Feature name	Parameter	Definition	Signal representations
Peak count		$n^{+}_{peak}(\mathbf{a})$	Autocorrelation
Neg-peak count		$n^{-}_{peak}(\mathbf{a})$	
Mean peak value		$\mu^{+}_{peak}(\mathbf{a}) = \mu(v^{+}_{peak}(\mathbf{a}))$	
Mean neg-peak value		$\mu^{-}_{peak}(\mathbf{a}) = \mu(v^{-}_{peak}(\mathbf{a}))$	
Std peak value		$\sigma^{+}_{peak}(\mathbf{a}) = \sigma(v^{+}_{peak}(\mathbf{a}))$	
Std neg-peak value		$\sigma^{-}_{peak}(\mathbf{a}) = \sigma(v^{-}_{peak}(\mathbf{a}))$	
Mean peak gap		$\mu(\Delta^{+}_{peak}(\mathbf{a}))$	
Mean neg-peak gap		$\mu(\Delta^{-}_{peak}(\mathbf{a}))$	
Std peak gap		$\sigma(\Delta^{+}_{peak}(\mathbf{a}))$	
Std neg-peak gap		$\sigma(\Delta^{-}_{peak}(\mathbf{a}))$	

Table A.2: Features extracted from autocorrelation.

Feature name	Parameter	Definition	Signal representations
Spectral energy max.		$\max\{\lvert f_1\rvert,...,\lvert f_{\frac{N}{2}}\rvert\}$	F.F.T
Freq. max.		$f_{max} = argmax\{\lvert f_1\rvert,...,\lvert f_{\frac{N}{2}}\rvert\}$	
Spectral energy median		$med\{\lvert f_1\rvert,...,\lvert f_{\frac{N}{2}}\rvert\}$	
Spectral energy mean		$\mu(\mathbf{f})$	
Spectral energy std		$\sigma(\mathbf{f})$	
Momentum	k	$M_k(\mathbf{f})$	
Nb-sup-threshold	v	$\sum_{i=1}^{N} \mathbb{1}\{\lvert f_i\rvert > v\}$	
Inf-threshold-ratio	v	$\frac{1}{N}\sum_{i=1}^{N} \mathbb{1}\{\lvert f_i\rvert < v\}$	
Inf-threshold-duration	v	$\Delta_t(\mathbf{f})$	
Percentile	α	$perc(\mathbf{f},\alpha)$	
Inter-percentile	α_1,α_2	$perc(\mathbf{f},\alpha_2) - perc(\mathbf{f},\alpha_1)$	
Log-mean-peak		$log(1 + \sum_{i=1}^{N} \lvert f_i\rvert \mathbb{1}\{\lvert f_i\rvert > v\})$	
Spectral energy max.		$\max\{\lvert W_{i,j}\rvert\}$	Spectrogram
Freq. max.		f_{max}	
Freq. bary.		f_{bary}	
Spectral energy median		Median of $\{\lvert W_{i,j}\rvert\}$	
Spectral energy mean		Mean of $\{\lvert W_{i,j}\rvert\}$	
Spectral energy std		Std of $\{\lvert W_{i,j}\rvert\}$	
Total energy		$\frac{2}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N/2} \lvert W_{i,j}\rvert^2$	
Total log-energy		$\frac{2}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N/2} log(1 + \lvert W_{i,j}\rvert^2)$	
Temp. width window-sup-thresh	v		
Freq. width window-sup-thresh	v	See 1.3.3 for details	
Gauss-window filtered energy	(a,b,θ)		

Table A.3: Features extracted from F.F.T and spectrograms.

B

Sensor relaxation

As described in Section 2 of Chapter 3, the piezo-electric sensor used by Tarkett works as a pool of free electrons that are displaced in function of the sensor deformation. So it can be compared to a capacitor and Section 3 explains through the *transfer function* notion how sensor's capacitance and the signal processing circuit can affect the signal on frequency aspects.

This also means that the sensor converges to an equilibrium state while sensor deformations stop, in the same way as the charge or discharge of a capacitor. This appendix shows a brief illustration that this sensor *relaxation* phenomenon follows the same patterns as a capacitor discharge with a very stable, precise and measurable parameter. Then this phenomenon is easily detectable and can give some valuable information in some situations.

In the *Walk DB* studied in Section 2.4 of Chapter 2, signals are recorded from a corridor with successive areas of sensor which makes relaxation phenomenon particularly observable.

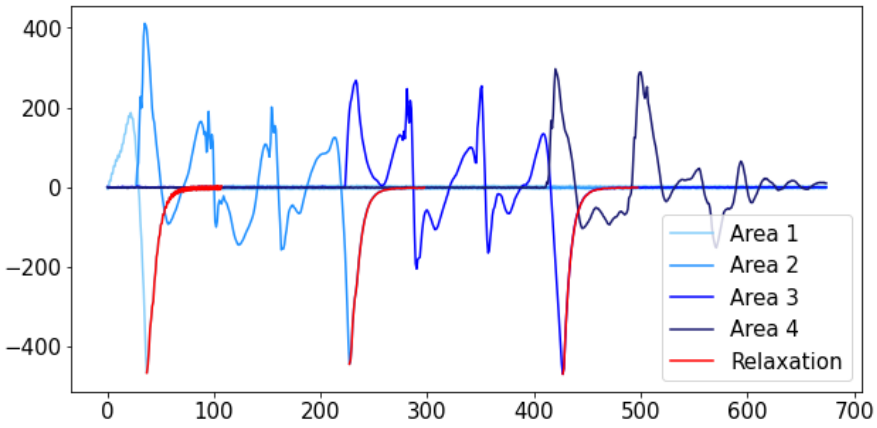


Figure B.1: Illustration of sensor relaxation on several areas of a corridor during a walk.

As illustrated in Figure B.1, while a moving object leaves the sensor area, a regular pattern is observable which corresponds to the sensor returning at its equilibrium state in terms of charges. This pattern is comparable to a capacitor charge/discharge of the form $Ke^{-\theta t}$ and is always the same. By measuring this θ parameter this behaviour is easily detectable.

Figures B.2 and B.3 shows various signals with a correlation value computed between a sliding window on the signal and the specific relaxation pattern. As this

phenomenon can be detected with a relatively simple approach, it could give some useful informations about daily life activities. For instance it happens while a person goes into the shower or on the bed if it is heavy enough (in this case floor's deformation is almost the same with or without a person on it). Moreover, estimating time spent in shower or on bed is interesting for elderly monitoring. So sensor's relaxation is a good example of simple and useful insight from physics.

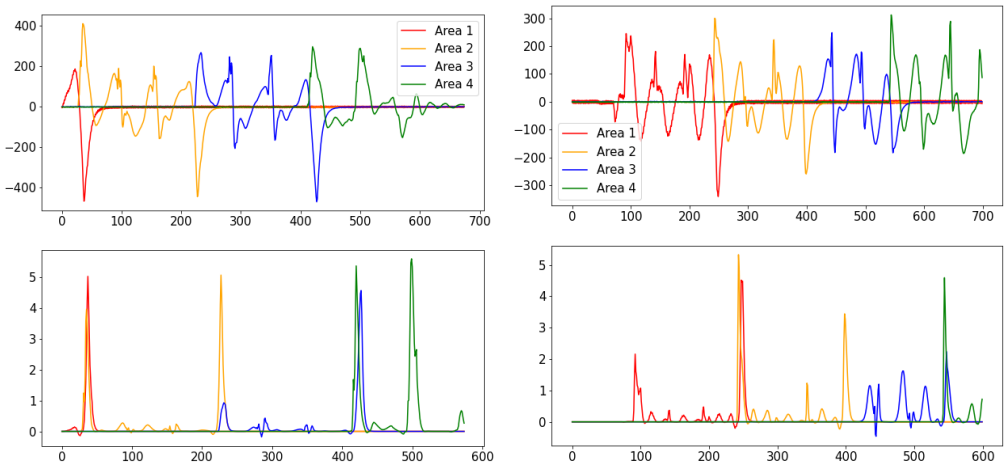


Figure B.2: Illustration of sensor relaxation on several areas of a corridor during a walk.

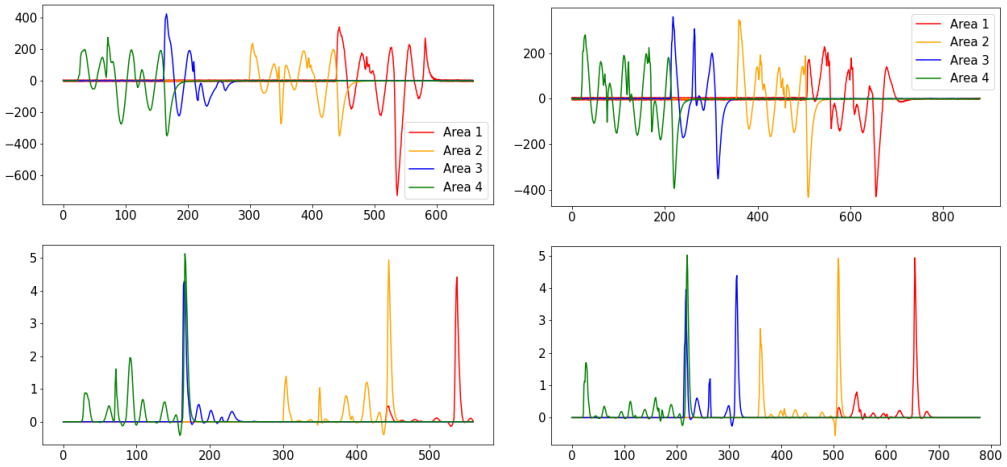


Figure B.3: Illustration of sensor relaxation on several areas of a corridor during a walk.

;

C

Additional results on transfer learning

In Chapter 3 and Section 5.4, we present our experiments and results on Tarkett fall detection data and public data, to evaluate several variants of *SER* and *STRUT* model-based transfer algorithms on decision trees. Here are presented additional results on synthetic and public data for the interested reader.

1 Synthetic data

As described in the corresponding section, synthetic experiments are done using Gaussian cluster distributions. In addition to class imbalance ratio, three kinds of parameters are involved to get a distribution shift between *source* and *target* domains. Clusters mean and variance are modified and some of them are completely re-drawn. Here are partial results according to the proportion of clusters that are re-drawn.

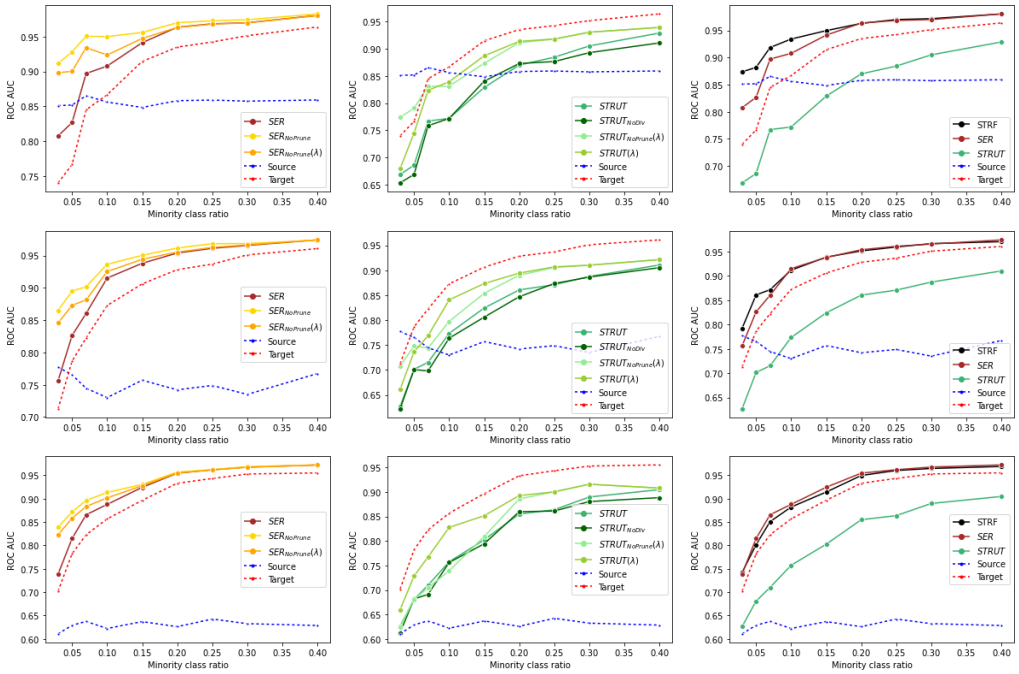


Figure C.1: Mean ROC AUC of transfer methods on synthetic data depending on the minority class ratio. First row : no cluster re-drawing. Second row : re-drawing one third of clusters. Third row : re-drawing two thirds of clusters.

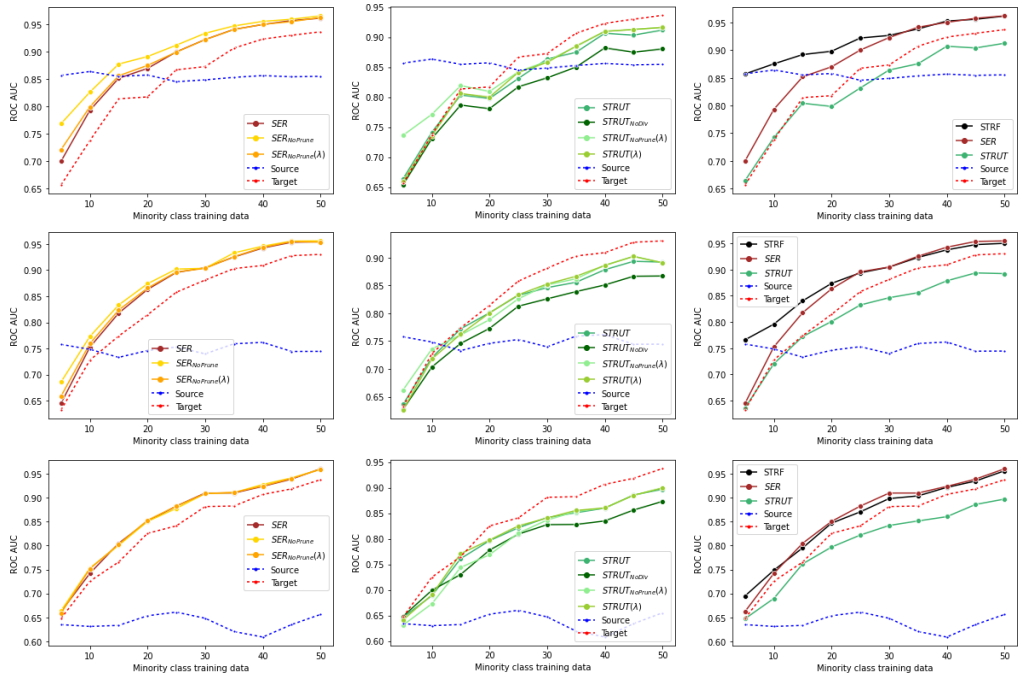


Figure C.2: Mean ROC AUC of transfer methods on synthetic data depending on the amount of target minority class data. First row : no cluster re-drawing. Second row : re-drawing one third of clusters. Third row : re-drawing two thirds of clusters.

2 Public data

Amazon reviews DB is composed by several domains corresponding to the type of products. Concerning the spam DB, data of 3 users are used as domains. In this appendix we present further results on transfer experiments on these databases than the ones exposed in Chapter 3.

Additional results on Amazon review DB concern experiments between other couples of domains and those on Spam DB concern the same couples of domains but with other directions of transfer (inverting source and target domains). *Source* and *target* domains are still used as references to give an idea of the necessity and the difficulty of each transfer learning task.

These results reinforce the statement that each *SER*/*STRUT* specific variant efficiency depends importantly on data and on the relations between source and target domains. Moreover they confirm the relevance of our *STRF* algorithm and its adaptability to various situations.

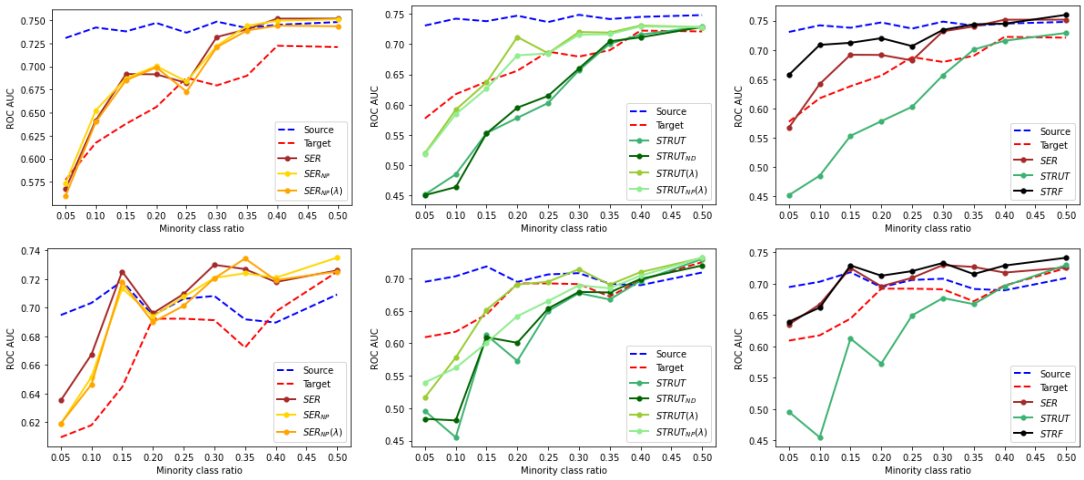


Figure C.3: Mean ROC AUC of transfer methods on Amazon reviews DB depending on the minority class ratio.
First row : from *home* products to *game* products. Second row : from *music* products to *game* products.

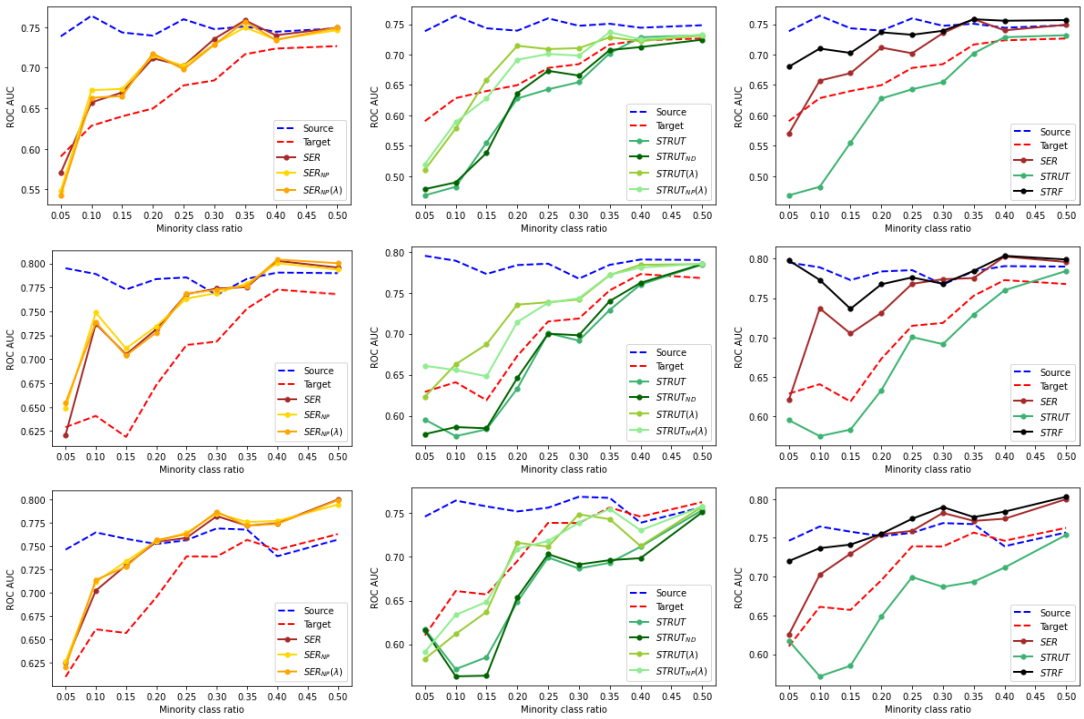


Figure C.4: Mean ROC AUC of transfer methods on Amazon reviews DB depending on the minority class ratio.
First row : from *phone* products to *game* products. Second row : from *phone* products to *home* products. Third row : from *phone* products to *music* products.

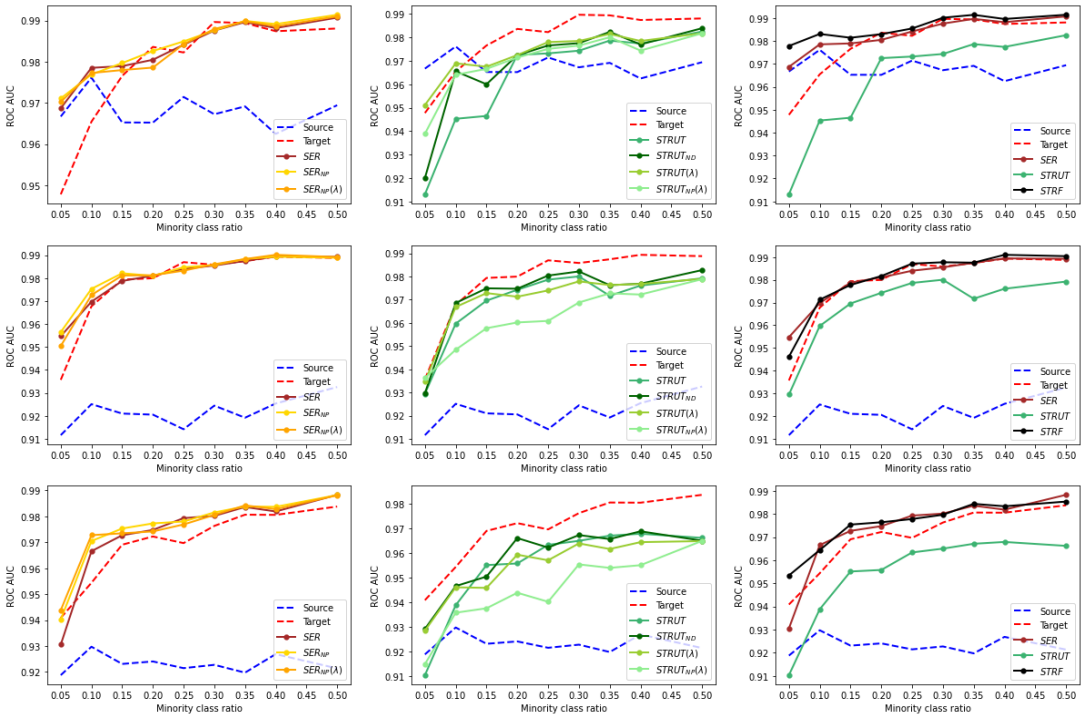


Figure C.5: Mean ROC AUC of transfer methods on Spam DB depending on the minority class ratio.
First row : from user "B" to user "A". Second row : from user "C" to user "A". Third row : from user "C" to user "B".

Bibliography

[1] Transfer algorithms on Decision Trees. https://github.com/atiqm/Transfer_DT, . [Online].

[2] Randomized equivalent decision trees. https://github.com/atiqm/equivalent_decision_trees, . [Online].

[3] Genetic pruning algorithm . https://github.com/atiqm/budget_learning/Gen/, . [Online].

[4] Synthetic Data Generator. https://github.com/SergioPeignier/TLSyntheticDataGenerator. [Online].

[5] Leonard A. Breslow and David W. Aha. Simplifying decision trees : A survey. *The Knowledge Engineering Review*, 12(1):1–40, 1997.

[6] Mubarak G. Abdu-Aguye and Walid Gomaa. Novel Approaches to Activity Recognition Based on Vector Autoregression and Wavelet Transforms. *Proceedings - 17th IEEE International Conference on Machine Learning and Applications, ICMLA 2018*, pages 951–954, 2019. doi: 10.1109/ICMLA.2018.00154.

[7] Umran Azziz Abdulla, Ken Taylor, Michael Barlow, and Khushnood Z. Naqshbandi. Measuring walking and running cadence using magnetometers. *Proceedings - 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2013*, pages 1458–1463, 2013. doi: 10.1109/TrustCom.2013.176.

[8] Hamid Abrishami Moghaddam and Amin Zare. Spatiotemporal wavelet correlogram for human action recognition. *International Journal of Multimedia Information Retrieval*, 2019. ISSN 2192662X. doi: 10.1007/s13735-018-00167-2. URL https://doi.org/10.1007/s13735-018-00167-2.

[9] Ali Akbari and Roozbeh Jafari. Transferring activity recognition models for new wearable sensors with deep generative domain adaptation. *IPSN 2019 - Proceedings of the 2019 Information Processing in Sensor Networks*, pages 85–96, 2019. doi: 10.1145/3302506.3310391.

[10] Samir Al-Stouhi and Chandan K. Reddy. Transfer learning for class imbalance problems with inadequate data. *Knowledge and Information Systems*, 48(1):201–228, 2016.

[11] Majd Alwan, Prabhu Jude Rajendran, Steve Kelli, David Mack, Siddharth Dalali, and Matt Wolfe I. A Smart and Passive Floor-Vibration Based Fall Detector for Elderly. 1:1003–1007, 2006.

[12] Atika Arshad, Sheroz Khan, A. H.M.Zahirul Alam, Rumana Tasnim, Teddy S. Gunawan, Robiah Ahmad, and Chandrasekharan Nataraj. An activity monitoring system for senior citizens living independently using capacitive sensing technique. *Conference Record - IEEE Instrumentation and Measurement Technology Conference*, 2016-July(August), 2016. ISSN 10915281. doi: 10.1109/I2MTC.2016.7520405.

[13] Moshe Babaioff, Nicole Immorlica, David Kempe, and Robert Kleinberg. A knapsack secretary problem with applications. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 4627 LNCS:16–28, 2007. ISSN 16113349. doi: 10.1007/978-3-540-74208-1_2.

[14] J Bala, J Huang, H Vafaie, K. DeJong, and H Wechsler. Hybrid learning using genetic algorithms and decision trees for pattern classification. *International Joint Conference on Artificial Intelligence*, 14(March):719–724, 1995. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.51.4794{&}rep=rep1{&}type=pdf.

[15] W. Banno and N. Shinomiya. Monitoring system for the elderly on staircase using passive rfid sensor tags. In *2019 IEEE 8th Global Conference on Consumer Electronics (GCCE)*, pages 816–817, 2019. doi: 10.1109/GCCE46687.2019.9015520.

[16] Ioannis Bargiotas, Argyris Kalogeratos, Myrto Limnios, Pierre Paul Vidal, Damien Ricard, and Nicolas Vayatis. Multivariate two-sample hypothesis testing through AUC maximization for biomedical applications. *ACM International Conference Proceeding Series*, pages 56–59, 2020. doi: 10.1145/3411408.3411422.

[17] Rodrigo Coelho Barros, Márcio Porto Basgalupp, André C.P.L.F. De Carvalho, and Alex A. Freitas. A survey of evolutionary algorithms for decision-tree induction. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, 42(3):291–312, 2011.

[18] Mohamed Bekkar, Hassiba Kheliouane Djemaa, and Taklit Akrouf Alitouche. Evaluation Measures for Models Assessment over Imbalanced Data Sets. *Journal of Information Engineering and Applications*, 3(10):27–38, 2011. ISSN 2225-0506.

[19] Shai Ben-David and Eli Dichterman. Learning with Restricted Focus of Attention. *Journal of Computer and System Sciences*, 56(3):277–298, 1998. ISSN 00220000. doi: 10.1006/jcss.1998.1569.

[20] Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. Analysis of representations for domain adaptation. *Advances in Neural Information Processing Systems*, pages 137–144, 2007. ISSN 10495258. doi: 10.7551/mitpress/7503.003.0022.

[21] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine Learning*, 79(1-2):151–175, 2010. ISSN 15730565. doi: 10.1007/s10994-009-5152-4.

[22] Shai Ben-David, Teresa Luu, Tyler Lu, and Dávid Pál. Impossibility theorems for domain adaptation. *Journal of Machine Learning Research*, 9:129–136, 2010. ISSN 15324435. doi: 10.1016/b978-1-78548-236-6.50004-0.

[23] Ganapati Bhat, Ranadeep Deb, Vatika Vardhan Chaurasia, Holly Shill, and Umit Y. Ogras. Online human activity recognition using low-power wearable devices. *IEEE/ACM International Conference on Computer-Aided Design, Digest of Technical Papers, ICCAD*, 2018. ISSN 10923152. doi: 10.1145/3240765.3240833.

[24] Yijun Bian and Huanhuan Chen. When Does Diversity Help Generalization in Classification Ensembles? *IEEE Transactions on Cybernetics*, pages 1–16, 2021.

[25] Gérard Biau, Erwan Scornet, and Johannes Welbl. Neural Random Forests. *Sankhya A*, 81(2):347–386, 2016.

[26] Steffen Bickel, Michael Brückner, and Tobias Scheffer. Discriminative learning for differing training and test distributions. *ACM International Conference Proceeding Series*, 227:81–88, 2007. doi: 10.1145/1273496.1273507.

[27] John Blitzer, Ryan McDonald, and Fernando Pereira. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 120–128. Association for Computational Linguistics, 2006.

[28] J Boudy, J L Baldinger, F Delavault, M Muller, I Farin, R V Andreao, S Torres-Mueller, A Serra, D Gaiti, F Rocaries, Ch Dietrich, A Lacombe, F Steenkeste, M Schaff, M Baer, A Ozguler, and S Vaysse. Telemedicine for elderly patient at home: the TelePat project. 19:74–81, 2006.

[29] A. K. Bourke and G. M. Lyons. A threshold-based fall-detection algorithm using a bi-axial gyroscope sensor. *Medical Engineering and Physics*, 30(1):84–90, 2008. ISSN 13504533. doi: 10.1016/j.medengphy.2006.12.001.

[30] A. K. Bourke, J. V. O'Brien, and G. M. Lyons. Evaluation of a threshold-based tri-axial accelerometer fall detection algorithm. *Gait and Posture*, 26(2):194–199, 2007. ISSN 09666362. doi: 10.1016/j.gaitpost.2006.09.012.

[31] Etienne Boursier and Vianney Perchet. SIC - MMAB : Synchronisation Involves Communication in Multiplayer Multi-Armed Bandits. *Advances in Neural Information Processing Systems*, 32, 2019.

[32] Etienne Boursier, Tristan Garrec, Vianney Perchet, and Marco Scarsini. Making the most of your day: online learning for optimal allocation of time. *Advances in Neural Information Processing Systems*, 34:11208–11219, 2021.

[33] Agata Brajdic and Robert Harle. Walk detection and step counting on unconstrained smartphones. page 225, 2013. doi: 10.1145/2493432.2493449.

[34] Max Bramer. Pre-pruning classification trees to reduce overfitting in noisy domains. pages 7–12, 2002.

[35] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[36] Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and regression trees*. Wadsworth, 1984.

[37] Stephanie A. Bridenbaugh and Reto W. Kressig. Laboratory review: The role of gait analysis in seniors' mobility and fall prevention. *Gerontology*, 57(3):256–264, 2011. ISSN 0304324X. doi: 10.1159/000322194.

[38] Damien Brulin, Yannick Benezeth, and Estelle Courtial. Posture recognition based on fuzzy logic for home monitoring of the elderly. *IEEE Transactions on Information Technology in Biomedicine*, 16(5):974–982, 2012. ISSN 10897771. doi: 10.1109/TITB.2012.2208757.

[39] Damien Brulin, Eric Campo, Clément Lejeune, and Daniel Estève. Time Slot Modeling of Life Habits in the Elderly for Decision-Making Support. *IRBM*, 41 (6):295–303, 2020.

[40] Randal E Bryant. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE TRANSACTIONS ON COMPUTERS*, C-35(8), 1986.

[41] E Campo and M Chan. Detecting Abnormal Behavior by Real-Time Monitoring of Patients. *AAAI Workshop on Automation as Caregiver*, pages 8–12, 2002.

[42] E. Campo, S. Bonhomme, M. Chan, and D. Esteve. Help to monitoring of elderly by using physical activities criteria. *Gerontechnology*, 7(2), 2008. ISSN 1569-1101. doi: 10.4017/gt.2008.07.02.021.00.

[43] Eric Campo, Damien Brulin, Yoann Charlon, and Elodie Bouzbib. *Activity recognition by classification method for weight variation measurement with an insole device for monitoring frail people*, volume 10898 LNCS. Springer International Publishing, 2018. ISBN 9783319945224. doi: 10.1007/978-3-319-94523-1_7. URL http://dx.doi.org/10.1007/978-3-319-94523-1{_}7.

[44] Nicoì Cesa-Bianchi, Shai Shalev-Shwartz, and Ohad Shamir. Efficient Learning with Partially Observed Attributes. *Journal of Machine Learning Research*, 12:2857–2878, 2011. URL http://www.jmlr.org/papers/volume12/cesa-bianchi11a/cesa-bianchi11a.pdf.

[45] Nicolo Cesa-Bianchi, Shai Shalev-shwartz, and Ohad Shamir. Some Impossibility Results for Budgeted Learning. *ICML Workshop on Budgeted Learning*, 2010.

[46] Sung-Hyuk Cha and Charles Tappert. A Genetic Algorithm for Constructing Compact Binary Decision Trees. *Journal of Pattern Recognition Research*, 4(1):1–13, 2013. doi: 10.13176/11.44.

[47] Kabalan Chaccour, Rony Darazi, Amir Hajjam El Hassani, and Emmanuel Andres. From Fall Detection to Fall Prevention: A Generic Classification of Fall-Related Systems. *IEEE Sensors Journal*, 17(3):812–822, 2017. ISSN 1530437X. doi: 10.1109/JSEN.2016.2628099.

[48] M. Chan, E. Campo, W. Bourennane, F. Bettahar, and Y. Charlon. Mobility behavior assessment using a smart-monitoring system to care for the elderly in a hospital environment. *ACM International Conference Proceeding Series*, 2014-May: 1–5, 2014. doi: 10.1145/2674396.2674397.

[49] Yee Seng Chan and Hwee Tou Ng. Estimating class priors in domain adaptation for word sense disambiguation. *In Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 89–96, 2006.

[50] Youngjae Chang, Akhil Mathur, Anton Isopoussu, Junehwa Song, and Fahim Kawsar. A systematic study of unsupervised domain adaptation for robust human-activity recognition. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 4(1):1–30, 2020. ISSN 24749567. doi: 10.1145/3380985.

[51] Jose M. Chaquet, Enrique J. Carmona, and Antonio Fernández-Caballero. A survey of video datasets for human action and activity recognition. *Computer Vision and Image Understanding*, 117(6):633–659, 2013. ISSN 1090235X. doi: 10.1016/j.cviu.2013.01.013. URL http://dx.doi.org/10.1016/j.cviu.2013.01.013.

[52] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[53] Jianfeng Chen, Jianmin Zhang, Alvin Harvey Kam, and Louis Shue. An automatic acoustic bathroom monitoring system. *Proceedings - IEEE International Symposium on Circuits and Systems*, pages 1750–1753, 2005. ISSN 02714310. doi: 10.1109/ISCAS.2005.1464946.

[54] Minmin Chen, Zhixiang Xu, Kilian Q. Weinberger, Olivier Chapelle, and Dor Kedem. Classifier Cascade for Minimizing Feature Evaluation Cost. *Proceedings of the 15th International Conference on Artificial Intelligence and Statistics (AISTATS)*, XX:218–226, 2012. ISSN 15337928.

[55] Stephan Clémençon, Marine Depecker, and Nicolas Vayatis. AUC optimization and the two-sample problem. *Advances in Neural Information Processing Systems 22 - Proceedings of the 2009 Conference*, pages 360–368, 2009.

[56] J R B Cockett and J A Herrera. Decision Tree Reduction. *Journal of the Association for Computing Machinery*, 37(4):815–842, 1990.

[57] J.R.B. COCKETT. *Discrete decision theory : manipulations*, volume 54. Elsevier, 1987.

[58] Patrick Connor and Arun Ross. Biometric recognition by gait: A survey of modalities and features. *Computer Vision and Image Understanding*, 167(June 2017):1–27, 2018. ISSN 1090235X. doi: 10.1016/j.cviu.2018.01.007. URL https://doi.org/10.1016/j.cviu.2018.01.007.

[59] Gabriella Contardo, Ludovic Denoyer, and Thierry Artières. Sequential cost-sensitive feature acquisition. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9897 LNCS: 284–294, 2016. ISSN 16113349. doi: 10.1007/978-3-319-46349-0_25.

[60] Diane Cook, Kyle D. Feuz, and Narayanan C. Krishnan. Transfer learning for activity recognition: A survey. *Knowledge and Information Systems*, 36(3):537–556, 2013. ISSN 02191377. doi: 10.1007/s10115-013-0665-3.

[61] Maria Cornacchia, Koray Ozcan, Yu Zheng, and Senem Velipasalar. A Survey on Activity Detection and Classification Using Wearable Sensors. *IEEE Sensors Journal*, 17(2):386–403, 2017. ISSN 1530437X. doi: 10.1109/JSEN.2016.2628346.

[62] Corinna Cortes, Mehryar Mohri, and Andrés Muñoz Medina. Adaptation algorithm and theory based on generalized discrepancy. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015-August(Dm):169–178, 2015. doi: 10.1145/2783258.2783368.

[63] Yang Qiang Dai Wenyuan, Xue Guirong, et al. Boosting for transfer learning. In *Proceedings of the 24th International Conference on Machine Learning, Corvallis, USA*, pages 193–200, 2007.

[64] Zoltán Daróczy. Generalized information functions. *Information and Control*, 16(1): 36–51, 1970.

[65] Jnaneshwar Das, Frédéric Py, Julio B.J. Harvey, John P. Ryan, Alyssa Gellene, Rishi Graham, David A. Caron, Kanna Rajan, and Gaurav S. Sukhatme. Data-driven robotic sampling for marine ecosystem monitoring. *International Journal of Robotics Research*, 34(12):1435–1452, 2015. ISSN 17413176. doi: 10.1177/0278364915587723.

[66] Hal Daumé III. Frustratingly easy domain adaptation. 2009.

[67] Oscar Day and Taghi M. Khoshgoftaar. A survey on heterogeneous transfer learning. *Journal of Big Data*, 4(1), 2017. ISSN 21961115. doi: 10.1186/s40537-017-0089-0.

[68] Pubali De, Amitava Chatterjee, and Anjan Rakshit. Recognition of Human Behavior for Assisted Living Using Dictionary Learning Approach. *IEEE Sensors Journal*, 18(6):2434–2441, 2018. ISSN 1530437X. doi: 10.1109/JSEN.2017.2787616.

[69] Kenneth De Jong and William M. Spears. Using Genetic Algorithms to Solve NP-Complete Problems. In *Internation Conference on Genetic Algorithms (ICGA)*, pages 124–132, 1989.

[70] Maria De Marsico and Alessio Mecca. A survey on gait recognition via wearable sensors. *ACM Computing Surveys*, 52(4), 2019. ISSN 15577341. doi: 10.1145/3340293.

[71] K. M. DeGoede and J. A. Ashton-Miller. Fall arrest strategy affects peak hand impact force in a forward fall. *Journal of Biomechanics*, 35(6):843–848, 2002. ISSN 00219290. doi: 10.1016/S0021-9290(02)00011-8.

[72] Yueng Santiago Delahoz and Miguel Angel Labrador. Survey on fall detection and fall prevention using wearable and external sensors. *Sensors (Switzerland)*, 14 (10):19806–19842, 2014. ISSN 14248220. doi: 10.3390/s141019806.

[73] Fani Deligianni, Yao Guo, and Guang Zhong Yang. From Emotions to Mood Disorders: A Survey on Gait Analysis Methodology. *IEEE Journal of Biomedical and Health Informatics*, 23(6):2302–2316, 2019. ISSN 21682208. doi: 10.1109/JBHI.2019.2938111.

[74] Houtao Deng, George Runger, Eugene Tuv, and Martyanov Vladimir. A time series forest for classification and feature extraction. *Information Sciences*, 239: 142–153, 2013.

[75] Kun Deng, Chris Bourke, Stephen Scott, Julie Sunderman, and Yaling Zheng. Bandit-based algorithms for budgeted learning. *Proceedings - IEEE International Conference on Data Mining, ICDM*, pages 463–468, 2007. ISSN 15504786. doi: 10.1109/ICDM.2007.91.

[76] Lazzaro Di Biase, Alessandro Di Santo, Maria Letizia Caminiti, Alfredo De Liso, Syed Ahmar Shah, Lorenzo Ricci, and Vincenzo Di Lazzaro. Gait analysis in parkinson's disease: An overview of the most accurate markers for diagnosis and symptoms monitoring. *Sensors (Switzerland)*, 20(12):1, 2020. ISSN 14248220. doi: 10.3390/s20123529.

[77] Chris Ding and Hanchuan Peng. Minimum redundancy feature selection from microarray gene expression data. *Journal of bioinformatics and computational biology*, 3(2):185–205, 2005.

[78] G. Diraco, A. Leone, and P. Siciliano. An active vision system for fall detection and posture recognition in elderly healthcare. pages 1536–1541, 2010.

[79] Pedro Domingos and Geoff Hulten. Mining High-Speed Data Streams. *Proceedings of The Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 71–80, 2000. ISSN 10844627. doi: 10.1145/347090.347107. URL http://dl.acm.org/citation.cfm?id=347107.

[80] Charalampos Doukas, Ilias Maglogiannis, Philippos Tragas, Dimitris Liapis, and Gregory Yovanof. Patient fall detection using support vector machines. In Christos Boukis, Aristodemos Pnevmatikakis, and Lazaros Polymenakos, editors, *Artificial Intelligence and Innovations 2007: from Theory to Applications*, pages 147–156, Boston, MA, 2007. Springer US. ISBN 978-0-387-74161-1.

[81] Thi Duong, Dinh Phung, Hung Bui, and Svetha Venkatesh. Efficient duration and hierarchical modeling for human activity recognition. *Artificial Intelligence*, 173(7-8):830–856, 2009. ISSN 00043702. doi: 10.1016/j.artint.2008.12.005. URL http://dx.doi.org/10.1016/j.artint.2008.12.005.

[82] Thi V. Duong, Hung H. Bui, Dinh Q. Phung, and Svetha Venkatesh. Activity Recognition and Abnormality Detection with the Switching Hidden Semi-Markov Model. *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, IEEE, Washington, D. C.*, pages 838–845, 2005.

[83] S. Durga, Rishabh Nag, and Esther Daniel. Survey on machine learning and deep learning algorithms used in internet of things (iot) healthcare. In *2019 3rd International Conference on Computing Methodologies and Communication (ICCMC)*, pages 1018–1022, 2019. doi: 10.1109/ICCMC.2019.8819806.

[84] Nashwa El-Bendary, Qing Tan, Frédérique C. Pivot, and Anthony Lam. Fall detection and prevention for the elderly: A review of trends and challenges.

International Journal on Smart Sensing and Intelligent Systems, 6(3):1230–1266, 2013. ISSN 11785608. doi: 10.21307/ijssis-2017-588.

[85] Charles Elkan. The foundations of cost-sensitive learning. *IJCAI International Joint Conference on Artificial Intelligence*, pages 973–978, 2001. ISSN 10450823.

[86] Baris Erol, Sevgi Z Gurbuz, and Moeness G Amin. GAN-based Synthetic Radar Micro-Doppler Augmentations for Improved Human Activity Recognition. pages 1–5, 2019.

[87] Daniel Evans, Jonathan Pester, Luis Vera, Donald Jeanmonod, and Rebecca Jeanmonod. Elderly fall patients triaged to the trauma bay: Age, injury patterns, and mortality risk. *American Journal of Emergency Medicine*, 33(11):1635–1638, 2015. ISSN 15328171. doi: 10.1016/j.ajem.2015.07.044. URL http://dx.doi.org/10.1016/j.ajem.2015.07.044.

[88] Meherwar Fatima and Maruf Pasha. Survey of Machine Learning Algorithms for Disease Diagnostic. *Journal of Intelligent Learning Systems and Applications*, 09(01): 1–16, 2017. ISSN 2150-8402. doi: 10.4236/jilsa.2017.91001.

[89] Basura Fernando, Amaury Habrard, Marc Sebban, and Tinne Tuytelaars. Unsupervised visual domain adaptation using subspace alignment. *Proceedings of the IEEE International Conference on Computer Vision*, pages 2960–2967, 2013.

[90] R Flamary, N Courty, D Tuia, and A Rakotomamonjy. Optimal Transport for Domain Adaptation. *IEEE Trans. Pattern Anal. Mach. Intell*, 1, 2016.

[91] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55 (1):119–139, 1997.

[92] Zhiwei Fu. An Innovative GA-Based Decision Tree Classifier in Large Scale Data Mining. pages 348–353, 1999.

[93] Gabriella CONTARDO. *Machine Learning under Resource Constraints*. PhD thesis, 2017.

[94] João Gama, Pedro Medas, and Pedro Rodrigues. Learning decision trees from dynamic data streams. (January 2005):573, 2005. doi: 10.1145/1066677.1066809.

[95] Tianshi Gao and Daphne Koller. Active Classification based on Value of Classifier. *Advances in neural information processing systems*, 24, 2011.

[96] Pascal Germain, Amaury Habrard, François Laviolette, and Emilie Morvant. PAC-Bayesian Theorems for Domain Adaptation with Specialization to Linear Classifiers. *arXiv preprint arXiv:1503.06944*, 2015. URL http://arxiv.org/abs/1503.06944.

[97] Pascal Germain, Amaury Habrard, François Laviolette, and Emilie Morvant. A new PAC-Bayesian perspective on domain adaptation. *33rd International Conference on Machine Learning, ICML 2016*, 2:1368–1379, 2016.

[98] Pascal Germain, Amaury Habrard, François Laviolette, and Emilie Morvant. PAC-Bayes and domain adaptation. *Neurocomputing*, 379:379–397, 2020. ISSN 18728286. doi: 10.1016/j.neucom.2019.10.105.

[99] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine Learning*, 63(1):3–42, 2006.

[100] Hemant Ghayvat, Muhammad Awais, Sharnil Pandya, Hao Ren, Saeed Akbarzadeh, Subhas Chandra Mukhopadhyay, Chen Chen, Prosanta Gope, Arpita Chouhan, and Wei Chen. Smart aging system: Uncovering the hidden wellness parameter for well-being monitoring and anomaly detection. *Sensors (Switzerland)*, 19(4), 2019. ISSN 14248220. doi: 10.3390/s19040766.

[101] Ashish Goel, Sanjeev Khanna, and Brad Null. The ratio index for budgeted learning, with applications. *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 18–27, 2009. doi: 10.1137/1.9781611973068.3.

[102] Heitor M. Gomes, Albert Bifet, Jesse Read, Jean Paul Barddal, Fabrício Enembreck, Bernhard Pfharinger, Geoff Holmes, and Talel Abdessalem. Adaptive random forests for evolving data stream classification. *Machine Learning*, 106(9-10):1469–1495, 2017. ISSN 15730565. doi: 10.1007/s10994-017-5642-8.

[103] Norberto A. Goussies, Sebastián Ubalde, and Marta Mejail. Transfer Learning Decision Forests for Gesture Recognition. *J. Mach. Learn. Res.*, 15(1):3667–3690, 2014.

[104] Russell Greiner, Adam J. Grove, and Dan Roth. Learning cost-sensitive active classifiers. *Artificial Intelligence*, 139(2):137–174, 2002. ISSN 00043702. doi: 10.1016/s0004-3702(02)00209-6.

[105] Arthur Gretton, Alex Smola, Jiayuan Huang, Marcel Schmittfull, Karsten Borgwardt, and Bernhard Schölkopf. Covariate Shift by Kernel Mean Matching. *Dataset Shift in Machine Learning*, pages 131–160, 2013. doi: 10.7551/mitpress/9780262170055.003.0008.

[106] Alexander Grubb and J. Andrew Bagnell. SpeedBoost: Anytime prediction with uniform near-optimality. *Journal of Machine Learning Research*, 22:458–466, 2012. ISSN 15337928.

[107] Quanquan Gu, Tong Zhang, Chris Ding, and Jiawei Han. Selective labeling via error bound minimization. *Advances in Neural Information Processing Systems*, 1:323–331, 2012. ISSN 10495258.

[108] Baofeng Guo and Mark S. Nixon. Gait feature subset selection by mutual information. *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans*, 39(1):36–46, 2009. ISSN 10834427. doi: 10.1109/TSMCA.2008.2007977.

[109] Ivring H. Lavalle and Peter C. Fishburn. Equivalent decision trees and their associated strategy sets. *Theory and Decision*, 23(1):37–63, 1987.

[110] Haixun Wang Wei Fan Philip S. Yu Jiawei Han. Mining Concept Drifting Data Streams using Ensemble Classifiers. pages 226–235, 2003.

[111] Thomas Hancock, Tao Jiang, Ming Li, and John Tromp. Lower Bounds on Learning Decision Lists and Trees. *Information and Computation*, 126(0040):114–122, 1996.

[112] Maayan Harel and Shie Mannor. Learning from multiple outlooks. *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*, pages 401–408, 2011.

[113] Mohammed Mehedi Hassan, Md Zia Uddin, Amr Mohamed, and Ahmad Almogren. A robust human activity recognition system using smartphone sensors and deep learning. *Future Generation Computer Systems*, 81:307–313, 2018. ISSN 0167739X. doi: 10.1016/j.future.2017.11.029. URL https://doi.org/10.1016/j.future.2017.11.029.

[114] Trevor Hastie, Robert Tibshirani, and Jerome H Friedman. *The elements of statistical learning II*.

[115] Jeffrey M Hausdorff. Gait dynamics, fractals and falls: finding meaning in the stride-to-stride fluctuations of human walking. *Human movement science*, 26(4):555–89, 2007. ISSN 0167-9457. doi: 10.1016/j.humov.2007.05.003. URL http://www.ncbi.nlm.nih.gov/pubmed/17618701{%}0Ahttp://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC2267927.

[116] Jeffrey M. Hausdorff, Dean A. Rios, and Helen K. Edelberg. Gait variability and fall risk in community-living older adults: A 1-year prospective study. *Archives of Physical Medicine and Rehabilitation*, 82(8):1050–1056, 2001. ISSN 00039993. doi: 10.1053/apmr.2001.24893.

[117] Haibo He and Edwardo A. Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, 2009.

[118] Zhen Yu He and Lian Wen Jin. Activity recognition from acceleration data using AR model representation and SVM. *Proceedings of the 7th International Conference on Machine Learning and Cybernetics, ICMLC*, 4(July):2245–2250, 2008. doi: 10.1109/ICMLC.2008.4620779.

[119] Zhenyu He. Activity recognition from accelerometer signals based on wavelet-AR model. *Proceedings of the 2010 IEEE International Conference on Progress in Informatics and Computing, PIC 2010*, 1:499–502, 2010. doi: 10.1109/PIC.2010.5687572.

[120] Rimminen Henry, Linnavuo Matti, and Sepponen Raimo. Human tracking using near field imaging. *Proceedings of the 2nd International Conference on Pervasive Computing Technologies for Healthcare 2008, PervasiveHealth*, pages 148–151, 2008. doi: 10.1109/PCTHEALTH.2008.4571055.

[121] Chen-Yu Hsu, Yuchen Liu, Zachary Kabelac, Rumen Hristov, Dina Katabi, and Christine Liu. Extracting Gait Velocity and Stride Length from Surrounding Radio Signals. pages 2116–2126, 2017. doi: 10.1145/3025453.3025937.

[122] Jiayuan Huang, Alexander J. Smola, Arthur Gretton, Karsten M. Borgwardt, and Bernhard Schölkopf. Correcting sample selection bias by unlabeled data. *Advances in Neural Information Processing Systems*, pages 601–608, 2007. ISSN 10495258. doi: 10.7551/mitpress/7503.003.0080.

[123] Zawar Hussain, Quan Z. Sheng, and Wei Emma Zhang. A review and categorization of techniques on device-free human activity recognition. *Journal of Network and Computer Applications*, 167(May):102738, 2020. ISSN 10958592. doi: 10.1016/j.jnca.2020.102738. URL https://doi.org/10.1016/j.jnca.2020.102738.

[124] Laurent Hyafil and Ronald L. Rivest. Constructing optimal binary decision trees is NP-Complete. *Information processing letters*, 5, 1976.

[125] Raul Igual, Carlos Medrano, and Inmaculada Plaza. Challenges, issues and trends in fall detection systems. *BioMedical Engineering Online*, 12(1), 2013. ISSN 1475925X. doi: 10.1186/1475-925X-12-66.

[126] Dariusz Jankowski and Konrad Jackowski. Evolutionary Algorithm for Decision Tree Induction. In *13th IFIP International Conference on Computer Information Systems and Industrial Management (CISIM)*, number November, pages 23–32, 2014. URL https://hal.inria.fr/hal-01405549.

[127] Salma Kammoun Jarray. Computer Vision Based Fall Detection Methods Using the Kinect Camera : A Survey. *International Journal of Computer Science and Information Technology*, 10(5):73–92, 2018. ISSN 09754660. doi: 10.5121/ijcsit.2018.10507.

[128] A. Y. Jeon, J. H. Kim, I. C. Kim, J. H. Jung, S. Y. Ye, J. H. Ro, S. H. Yoon, J. M. Son, B. C. Kim, B. J. Shin, and G. R. Jeon. Implementation of the personal emergency response system using a 3-axial accelerometer. *Proceedings of the IEEE/EMBS Region 8 International Conference on Information Technology Applications in Biomedicine, ITAB*, 00:223–226, 2007. doi: 10.1109/ITAB.2007.4407387.

[129] Hongbo Jiang, Chao Cai, Xiaoqiang Ma, Yang Yang, and Jiangchuan Liu. Smart Home Based on WiFi Sensing: A Survey. *IEEE Access*, 6:13317–13325, 2018. ISSN 21693536. doi: 10.1109/ACCESS.2018.2812887.

[130] Michael I Jordan, Zoubin Ghahramani, and Lawrence K Saul. Hidden Markov Decision Trees. *Advances in neural information processing systems*, 1996.

[131] Sakari Junnila, Alireza Akhbardeh, Alpo Värri, and Teemu Koivistoinen. An EMFi-film sensor based ballistocardiographic chair: Performance and cycle extraction method. *IEEE Workshop on Signal Processing Systems, SiPS: Design and Implementation*, 2005:373–377, 2005. ISSN 15206130. doi: 10.1109/SIPS.2005.1579896.

[132] Sakari Junnila, Alireza Akhbardeh, and Alpo Värri. An electromechanical film sensor based wireless ballistocardiographic chair: Implementation and performance. *Journal of Signal Processing Systems*, 57(3):305–320, 2009. ISSN 19398018. doi: 10.1007/s11265-008-0307-2.

[133] Mohammad Kachuee, Orpaz Goldstein, Kimmo Karkkainen, Sajad Darabi, and Majid Sarrafzadeh. Opportunistic learning: Budgeted cost-sensitive learning from data streams. *arXiv preprint arXiv:1901.00243*, 2019.

[134] Aloak Kapoor and Russell Greiner. Learning and classifying under hard budgets. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3720 LNAI:170–181, 2005. ISSN 03029743. doi: 10.1007/11564096_20.

[135] Sergey Karayev, Mario Fritz, and Trevor Darrell. Anytime recognition of objects and scenes. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 572–579, 2014. ISSN 10636919. doi: 10.1109/CVPR.2014.80.

[136] K Kargupta, B-H Park, and Haimonti Dutta. *IEEE Transactions on Knowledge and Data Engineering*, 18(8):1028–1042, 2006.

[137] S. Keskar and R. Banerjee. Time-recurrent HMM decision tree to generate alerts for heart-guard wearable computer. *Computing in Cardiology*, 38:605–608, 2011. ISSN 23258861 2325887X.

[138] Shehroz S. Khan and Jesse Hoey. Review of fall detection techniques: A data availability perspective. *Medical Engineering and Physics*, 39:12–22, 2017. ISSN 18734030. doi: 10.1016/j.medengphy.2016.10.014.

[139] Dae Sun Kim, Yeul Min Baek, and Whoi Yul Kim. Reducing overfitting of adaboost by clustering-based pruning of hard examples. *Proceedings of the 7th International Conference on Ubiquitous Information Management and Communication, ICUIMC 2013*, pages 4–6, 2013. doi: 10.1145/2448556.2448646.

[140] In Su Kim, Hong Seok Choi, Kwang Moo Yi, Jin Young Choi, and Seong G. Kong. Intelligent visual surveillance - A survey. *International Journal of Control, Automation and Systems*, 8(5):926–939, 2010. ISSN 15986446. doi: 10.1007/s12555-010-0501-4.

[141] Jeremy Z. Kolter and Marcus A. Maloof. Dynamic weighted majority: A new ensemble method for tracking concept drift. *Proceedings - IEEE International Conference on Data Mining, ICDM*, pages 123–130, 2003. ISSN 15504786. doi: 10.1109/icdm.2003.1250911.

[142] S. B. Kotsiantis. Feature selection for machine learning classification problems: A recent overview. *Artificial Intelligence Review*, 42(1):157, 2014. ISSN 02692821. doi: 10.1007/s10462-011-9230-1.

[143] C. Krier, D. François, F. Rossi, and M. Verleysen. Feature clustering and mutual information for the selection of variables in spectral data. *ESANN 2007 Proceedings - 15th European Symposium on Artificial Neural Networks*, (April):157–162, 2007.

[144] Miroslav Kubat and Stan Matwin. Addressing the Curse of Imbalanced Training Sets: One-Sided Selection. *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 179–186, 1997. URL http://sci2s.ugr.es/keel/pdf/algorithm/congreso/kubat97addressing.pdf.

[145] Swarn Avinash Kumar, Harsh Kumar, Vishal Dutt, and Pooja Dixit. The Role of Machine Learning in COVID-19 in Medical Domain: A Survey. *Journal on Recent Innovation in Cloud Computing, Virtualization & Web Applications*, 4(1):1–12, 2020.

[146] L. I. Kuncheva and C. J. Whitaker. Ten measures of diversity in classifier ensembles: Limits for two classifiers. *IEE Colloquium (Digest)*, (50):73–82, 2001. ISSN 09633308. doi: 10.1049/ic:20010105.

[147] S. Y. Kung. Feature selection. *Kernel Methods and Machine Learning*, 1:118–138, 2014. doi: 10.1017/CBO9781139176224.007.

[148] Kai Kunze, Gernot Bahle, Paul Lukowicz, and Kurt Partridge. Can magnetic field sensors replace gyroscopes in wearable sensing applications? *Proceedings - International Symposium on Wearable Computers, ISWC*, pages 3–6, 2010. ISSN 15504816. doi: 10.1109/ISWC.2010.5665859.

[149] Ilja Kuzborskij. *Theory and Algorithms for Hypothesis Transfer Learning*. PhD thesis, 2018.

[150] Douglas D. Larish, Philip E. Martin, and Michael Mungiole. Characteristic Patterns of Gait in the Healthy Old. *Annals of the New York Academy of Sciences*, 515(1):18–32, 1988.

[151] Jun Won Lee and Christophe Giraud-Carrier. Transfer learning in decision trees. In *IEEE International Conference on Neural Networks - Conference Proceedings*, volume 1, pages 726–731, 2007. ISBN 142441380X. doi: 10.1109/IJCNN.2007.4371047.

[152] Qiang Li, John A. Stankovic, Mark A. Hanson, Adam T. Barth, John Lach, and Gang Zhou. Accurate, fast fall detection using gyroscopes and accelerometer-derived posture information. *Proceedings - 2009 6th International Workshop on Wearable and Implantable Body Sensor Networks, BSN 2009*, pages 138–143, 2009. doi: 10.1109/BSN.2009.46.

[153] Xinyu Li, Xiaojun Jing, and Yuan He. Unsupervised Domain Adaptation for Human Activity Recognition in Radar. *IEEE National Radar Conference - Proceedings*, 2020-Septe:623–630, 2020. ISSN 10975659. doi: 10.1109/RadarConf2043947.2020.9266433.

[154] Yun Li, K.C. Ho, and Mihail Popescu. A microphone array system for automatic fall detection. *IEEE Transactions on Biomedical Engineering*, 59(2):1291–1301, 2012. URL http://ovidsp.ovid.com/ovidweb.cgi?T=JS{&}PAGE=reference{&}D=emed14{&}NEWS=N{&}AN=364706303.

[155] Chunquan Liang, Yang Zhang, and Qun Song. Decision Tree for Dynamic and Uncertain Data Streams. *2nd Asian Conference on Machine Learning*, 13:209–224, 2010. ISSN 15324435. URL http://www.ece.unm.edu/{~}jimp/codesign/papers/LiangDTDynamicUncertain.pdf.

[156] Kang Li Liang Ge, Jing Gao, Hung Ngo and Aidong Zhang Computer. On Handling Negative Transfer and Imbalanced Distributions in Multiple Source Transfer Learning. (December 2013), 2010. ISSN 09574174. doi: 10.1002/sam. URL http://arxiv.org/abs/1010.4784.

[157] Yong Gyu Lim, Ki Hwan Hong, Ko Keun Kim, Jae Hyuk Shin, Seung Min Lee, Gih Sung Chung, Hyun Jae Baek, Do Un Jeong, and Kwang Suk Park. Monitoring physiological signals using nonintrusive sensors installed in daily life equipment. *Biomedical Engineering Letters*, 1(1):11–20, 2011. ISSN 20939868. doi: 10.1007/s13534-011-0012-0.

[158] Wesllen Sousa Lima, Eduardo Souto, Khalil El-Khatib, Roozbeh Jalali, and Joao Gama. Human activity recognition using inertial sensors in a smartphone: An overview. *Sensors (Switzerland)*, 19(14):14–16, 2019. ISSN 14248220. doi: 10.3390/s19143213.

[159] Li Liu, Yuxin Peng, Shu Wang, Ming Liu, and Zigang Huang. Complex activity recognition using time series pattern dictionary learned from ubiquitous sensors. *Information Sciences*, 340-341:41–57, 2016. ISSN 00200255. doi: 10.1016/j.ins.2016.01.020. URL http://dx.doi.org/10.1016/j.ins.2016.01.020.

[160] Xiaobo Liu, Zhentao Liu, Guangjun Wang, Zhihua Cai, and Harry Zhang. Ensemble Transfer Learning Algorithm. *IEEE Access*, 6:2389–2396, 2017. ISSN 21693536. doi: 10.1109/ACCESS.2017.2782884.

[161] Zhigang Liu, Yanan Song, Ye Shang, and Jinkuan Wang. Posture recognition algorithm for the elderly based on BP neural networks. *Proceedings of the 2015 27th Chinese Control and Decision Conference, CCDC 2015*, pages 1446–1449, 2015. doi: 10.1109/CCDC.2015.7162146.

[162] Shane A. Lowe and Gearóid ÓLaighin. Monitoring human health behaviour in one's living environment: A technological review. *Medical Engineering and Physics*, 36(2):147–168, 2014. ISSN 13504533. doi: 10.1016/j.medengphy.2013.11.010.

[163] Wenhao Luo, Changjoo Nam, and Katia Sycara. Online decision making for stream-based robotic sampling via submodular optimization. *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, 2017-Novem: 118–123, 2017. doi: 10.1109/MFI.2017.8170416.

[164] Omid Madani, Daniel J Lizotte, and Russell Greiner. Budgeted learning, Part I: The multi-armed bandit case. Technical report, Citeseer, 2003. URL http://scholar.google.com/scholar?hl=en{&}btnG=Search{&}q=intitle:Budgeted+Learning,+Part+I:+The+Multi-Armed+Bandit+Case{#}0.

[165] Behrooz Mahasseni, Sinisa Todorovic, and Alan Fern. Budget-aware deep semantic video segmentation. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-Janua:2077–2086, 2017. doi: 10.1109/CVPR.2017.224.

[166] Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain adaptation: Learning bounds and algorithms. *COLT 2009 - The 22nd Conference on Learning Theory*, (2007), 2009.

[167] Diego Marron, Albert Bifet, and Gianmarco De Francisci Morales. Random forests of very fast decision trees on GPU for mining evolving big data streams. *Frontiers in Artificial Intelligence and Applications*, 263:615–620, 2014. ISSN 09226389. doi: 10.3233/978-1-61499-419-0-615.

[168] Hamid Medjahed, Dan Istrate, Jérôme Boudy, Jean-louis Baldinger, Bernadette Dorizzi, Imad Belfeki, Vinicius Martins, François Steenkeste, and Rodrigo Andreao. A multimodal platform for database recording and elderly people monitoring. *International Conference on Bio-inspired Systems and Signal Processing*, pages pp.385 – 392, 2008. doi: 10.5220/0001065803850392.

[169] Hamid Medjahed, Dan Istrate, Jerome Boudy, and Bernadette Dorizzi. Human activities of daily living recognition using fuzzy logic for elderly home monitoring. *IEEE International Conference on Fuzzy Systems*, pages pp.2001 – 2006, 2009. doi: 10.1109/FUZZY.2009.5277257.

[170] Hamid Medjahed, Dan Istrate, Jerome Boudy, Jean-louis Baldinger, and Bernadette Dorizzi. A pervasive multi-sensor data fusion for smart home healthcare monitoring. *IEEE International Conference on Fuzzy Systems*, pages 1466–1473, 2011. doi: 10.1109/FUZZY.2011.6007636.

[171] Lee Middleton, Alex A. Buss, Alex Bazin, and Mark S. Nixon. A floor sensor system for gait recognition. *Proceedings - Fourth IEEE Workshop on Automatic Identification Advanced Technologies, AUTO ID 2005*, 2005:171–180, 2005. doi: 10.1109/AUTOID.2005.2.

[172] Ludovic Minvielle. *Classification d ' événements à partir de capteurs sol – application au suivi de personnes fragiles*. PhD thesis, 2020.

[173] Ludovic Minvielle and Julien Audiffren. Nursenet: Monitoring elderly levels of activity with a piezoelectric floor. *Sensors (Switzerland)*, 19(18), 2019. ISSN 14248220. doi: 10.3390/s19183851.

[174] Ludovic Minvielle, Mounir Atiq, Renan Serra, Mathilde Mougeot, and Nicolas Vayatis. Fall detection using smart floor sensor and supervised learning. pages 3445–3448, July 2017.

[175] Ludovic Minvielle, Mounir Atiq, Renan Serra, Mathilde Mougeot, and Nicolas Vayatis. Fall detection using smart floor sensor and supervised learning. In *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 3445–3448. IEEE, 2017.

[176] Ihab Mohammed, Shadha Tabatabai, Ala Al-Fuqaha, Faissal El Bouanani, Junaid Qadir, Basheer Qolomany, and Mohsen Guizani. Budgeted online selection of candidate iot clients to participate in federated learning. *IEEE Internet of Things Journal*, 8(7):5938–5952, 2021. ISSN 23274662. doi: 10.1109/JIOT.2020.3036157.

[177] Bernard M E Moret. Decision Trees and Diagrams. *Computing Surveys*, 14(593-623), 1982.

[178] Emilie Morvant, Amaury Habrard, and Stéphane Ayache. Parsimonious unsupervised and semi-supervised domain adaptation with good similarity functions. *Knowledge and Information Systems*, 33(2):309–349, 2012. ISSN 02191377. doi: 10.1007/s10115-012-0516-7.

[179] Subhas Chandra Mukhopadhyay. Activity and Anomaly Detection in Smart Home: A Survey. *Next Generation Sensors and Systems*, 16:1–330, 2015. doi: 10.1007/978-3-319-21671-3.

[180] S Mulroy, J Gronley, W Weiss, C Newsam, and J Perry. Use of cluster analysis for gait pattern classification of patients in the early and late recovery phases following stroke. *Gait and Posture*, 18(1):114–125, 2003. URL http://ovidsp.ovid.com/ovidweb.cgi?T=JS{&}PAGE=reference{&}D=emed6{&}NEWS=N{&}AN=2003274911.

[181] Alvaro Muro-de-la Herran, Begoña García-Zapirain, and Amaia Méndez-Zorrilla. Gait analysis methods: An overview of wearable and non-wearable systems, highlighting clinical applications. *Sensors (Switzerland)*, 14(2):3362–3394, 2014. ISSN 14248220. doi: 10.3390/s140203362.

[182] Feng Nan and Venkatesh Saligrama. Adaptive classification for prediction under a budget. In *Advances in Neural Information Processing Systems*, volume 2017-Decem, pages 4728–4738, 2017.

[183] Feng Nan, Joseph Wang, and Venkatesh Saligrama. Feature-budgeted random forest. *32nd International Conference on Machine Learning, ICML 2015*, 3:1983–1991, 2015.

[184] Feng Nan, Joseph Wang, and Venkatesh Saligrama. Pruning random forests for prediction on a budget. *Advances in Neural Information Processing Systems*, pages 2342–2350, 2016. ISSN 10495258. URL http://arxiv.org/abs/1606.05060.

[185] Alexander Neil B. Gait disorders in older adults. 1996.

[186] Nam T. Nguyen, Svetha Venkatesh, Geoff A.W. West, and Hung H. Bui. Learning people movement model from multiple cameras for behaviour recognition. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3138:315–324, 2004. ISSN 16113349. doi: 10.1007/978-3-540-27868-9_33.

[187] Marek Novák, Miroslav Biňas, and František Jakab. Unobtrusive anomaly detection in presence of elderly in a smart-home environment. *Proceedings of 9th International Conference, ELEKTRO 2012*, (September 2016):341–344, 2012. doi: 10.1109/ELEKTRO.2012.6225617.

[188] Nikunj Chandrakant Oza and Stuart Russell. *Online ensemble learning*. University of California, Berkeley, 2001.

[189] Mika Paajanen, Jukka Lekkala, and Kari Kirjavainen. ElectroMechanical Film (EMFi) - a new multipurpose electret material. *Sensors and Actuators, A: Physical*, 84(1-2):95–102, 2000.

[190] Artidoro Pagnoni, Stefan Gramatovici, and Samuel Liu. PAC Learning Guarantees Under Covariate Shift. *arXiv preprint arXiv:1812.06393*, 2018.

[191] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, Oct 2010.

[192] Natthapon Pannurat, Surapa Thiemjarus, and Ekawit Nantajeewarawat. Automatic fall monitoring: A review. *Sensors (Switzerland)*, 14(7):12900–12936, 2014. ISSN 14248220. doi: 10.3390/s140712900.

[193] Hanchuan Peng, Fuhui Long, and C. Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1226–1238, 2005. doi: 10.1109/TPAMI.2005.159.

[194] Frank Perbet, Bjorn Stenger, and Atsuto Maki. Random Forest Clustering and Application to Video Segmentation. pages 1–10, 2012.

[195] James T. Perry, Scott Kellog, Sundar M. Vaidya, Jong Hoon Youn, Hesham Ali, and Hamid Sharif. Survey and evaluation of real-time fall detection approaches. *6th International Symposium on High Capacity Optical Networks and Enabling Technologies, HONET '09*, (May 2014):158–164, 2009. doi: 10.1109/HONET.2009.5423081.

[196] Sven Peter, Ferran Diego, Fred A. Hamprecht, and Boaz Nadler. Cost efficient gradient boosting. *Advances in Neural Information Processing Systems*, 2017-Decem (Nips 2017):1552–1562, 2017. ISSN 10495258.

[197] Kate Van Poppel, Stephen P. Fulton, Amy McGregor, Michelle Ellis, Andrea Patters, and James Wheless. Prospective study of the Emfit movement monitor. *Journal of Child Neurology*, 28(11):1434–1436, 2013. ISSN 08830738. doi: 10.1177/0883073812471858.

[198] Rafael Possas, Sheila Pinto Caceres, and Fabio Ramos. Egocentric Activity Recognition on a Budget. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 5967–5976, 2018. doi: 10.1109/CVPR.2018.00625.

[199] Octavian A. Postolache, Pedro M.B.Silva Girao, Joaquim Mendes, Eduardo C. Pinheiro, and Gabriela Postolache. Physiological parameters measurement based on wheelchair embedded sensors and advanced signal processing. *IEEE Transactions on Instrumentation and Measurement*, 59(10):2564–2574, 2010. ISSN 00189456. doi: 10.1109/TIM.2010.2057590.

[200] Aria Ghora Prabono, Bernardo Nugroho Yahya, and Seok Lyong Lee. Hybrid domain adaptation with deep network architecture for end-to-end cross-domain human activity recognition. *Computers and Industrial Engineering*, (November): 106953, 2020. ISSN 03608352. doi: 10.1016/j.cie.2020.106953. URL https://doi.org/10.1016/j.cie.2020.106953.

[201] Peter Prettenhofer and Benno Stein. Cross-Lingual Adaptation using Structural Correspondence Learning. (July):1118–1127, 2010. ISSN 21576904. doi: 10.1145/2036264.2036277. URL http://arxiv.org/abs/1008.0716.

[202] Adnan Qayyum, Junaid Qadir, Muhammad Bilal, and Ala Al-Fuqaha. Secure and Robust Machine Learning for Healthcare: A Survey. *IEEE Reviews in Biomedical Engineering*, 14:156–180, 2021. ISSN 19411189. doi: 10.1109/RBME.2020.3013489.

[203] Tsirizo Rabenoro, Jerome Lacaille, Marie Cottrell, and Fabrice Rossi. Anomaly detection based on indicators aggregation. *Proceedings of the International Joint Conference on Neural Networks*, pages 2548–2555, 2014. doi: 10.1109/IJCNN.2014.6889841.

[204] Parisa Rashidi and Diane J. Cook. Activity knowledge transfer in smart environments. *Pervasive and Mobile Computing*, 7(3):331–343, 2011. ISSN 15741192. doi: 10.1016/j.pmcj.2011.02.007. URL http://dx.doi.org/10.1016/j.pmcj.2011.02.007.

[205] Ievgen Redko, Amaury Habrard, and Marc Sebban. Theoretical Analysis of Domain Adaptation with Optimal Transport. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10535 LNAI:737–753, 2017. ISSN 16113349. doi: 10.1007/978-3-319-71246-8_45.

[206] Ievgen Redko, Amaury Habrard, Emilie Morvant, Marc Sebban, and Younès Bennani. *Advances in domain adaption theory*. 2019. ISBN 9781785482366. doi: 10.1016/C2016-0-05108-2.

[207] Tuomas Reinvuo, Manne Hannula, Hannu Sorvoja, Esko Alasaarela, and Risto Myllylä. Measurement of respiratory rate with high-resolution accelerometer and EMFit pressure sensor. *Proceedings of the 2006 IEEE Sensors Applications Symposium*, (February):192–195, 2006. doi: 10.1109/sas.2006.1634270.

[208] Lingmei Ren and Yanjun Peng. Research of fall detection and fall prevention technologies: A systematic review. *IEEE Access*, 7:77702–77722, 2019. ISSN 21693536. doi: 10.1109/ACCESS.2019.2922708.

[209] Henry Rimminen, Juha Lindström, Matti Linnavuo, and Raimo Sepponen. Detection of falls among the elderly by a floor sensor using the electric near field. *IEEE Transactions on Information Technology in Biomedicine*, 14(6):1475–1476, 2010. ISSN 10897771. doi: 10.1109/TITB.2010.2051956.

[210] Laurence Z. Rubenstein. Falls in older people: Epidemiology, risk factors and strategies for prevention. *Age and Ageing*, 35(SUPPL.2):37–41, 2006. ISSN 00020729. doi: 10.1093/ageing/afl084.

[211] Dymitr Ruta and Bogdan Gabrys. Application of the evolutionary algorithms for classifier selection in multiple classifier systems with majority voting. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2096:399–408, 2001. ISSN 16113349. doi: 10.1007/3-540-48219-9_40.

[212] Amir Saffari, Christian Leistner, Jakob Santner, Martin Godec, and Horst Bischof. On-line random forests. *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops 2009*, pages 1393–1400, 2009. ISSN 08856125. doi: 10.1109/ICCVW.2009.5457447.

[213] Jeremie Saives, Clement Pianon, and Gregory Faraut. Activity Discovery and Detection of Behavioral Deviations of an Inhabitant from Binary Sensors. *IEEE Transactions on Automation Science and Engineering*, 12(4):1211–1224, 2015. ISSN 15455955. doi: 10.1109/TASE.2015.2471842.

[214] Andrea Rosales Sanabria and Juan Ye. Unsupervised domain adaptation for activity recognition across heterogeneous datasets. *Pervasive and Mobile Computing*, 64:101147, 2020. ISSN 15741192. doi: 10.1016/j.pmcj.2020.101147. URL https://doi.org/10.1016/j.pmcj.2020.101147.

[215] Cullen Schaffer. Overfitting Avoidance as Bias. *Machine Learning*, 10(2):153–178, 1993.

[216] Noam Segev, Maayan Harel, Shie Mannor, Koby Crammer, and Ran El-Yaniv. Learn on source, refine on target: a model transfer learning framework with random forests. *IEEE transactions on pattern analysis and machine intelligence*, 39(9): 1811–1824, 2017.

[217] Renan Serra. *Développement et caractérisation d'un système de sol piézoélectrique intelligent. Application à la détection des chutes.* PhD thesis, Université de Strasbourg, 2017.

[218] Renan Serra, Dominique Knittel, Pascal Di Croce, and Richard Peres. Activity Recognition with Smart Polymer Floor Sensor: Application to Human Footstep Recognition. *IEEE Sensors Journal*, 16(14):5757–5775, 2016. ISSN 1530437X. doi: 10.1109/JSEN.2016.2554360.

[219] Qiongfeng Shi, Zixuan Zhang, Tianyiyi He, Zhongda Sun, Bingjie Wang, Yuqin Feng, Xuechuan Shan, Budiman Salam, and Chengkuo Lee. Deep learning enabled smart mats as a scalable floor monitoring system. *Nature Communications*, pages 1–11, 2020. ISSN 2041-1723. doi: 10.1038/s41467-020-18471-z. URL http://dx.doi.org/10.1038/s41467-020-18471-z.

[220] Xiaoran Shi, Yaxin Li, Feng Zhou, and Lei Liu. Human Activity Recognition Based on Deep Learning Method. In *2018 International Conference on Radar*, pages 1–5. IEEE, 2018.

[221] Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90(2):227–244, 2000. ISSN 03783758. doi: 10.1016/s0378-3758(00)00115-4.

[222] Muhammad Shoaib, Stephan Bosch, Ozlem Durmaz Incel, Hans Scholten, and Paul J.M. Havinga. *Fusion of smartphone motion sensors for physical activity recognition*, volume 14. 2014. ISBN 1014610176. doi: 10.3390/s140610146.

[223] Anuradha Singh, Saeed Ur Rehman, Sira Yongchareon, and Peter Han Joo Chong. Sensor Technologies for Fall Detection Systems: A Review. *IEEE Sensors Journal*, 20(13):6889–6919, 2020. ISSN 15581748. doi: 10.1109/JSEN.2020.2976554.

[224] Jasvinder Pal Singh, Sanjeev Jain, Sakshi Arora, and Uday Pratap Singh. Vision-based gait recognition: A survey. *IEEE Access*, 6:70497–70527, 2018.

[225] Evaggelos Spyrou, Eirini Mathe, Georgios Pikramenos, Konstantinos Kechagias, and Phivos Mylonas. Data Augmentation vs. Domain Adaptation—A Case Study in Human Activity Recognition. *Technologies*, 8(4):55, 2020.

[226] Lewis Sudarsky. Geriatrics: Gait disorders in the elderly. *The New English Journal of medicine*, 323(16):1120–1123, 1990.

[227] Masashi Sugiyama, Shinichi Nakajima, Hisashi Kashima, Paul Von Bünau, and Motoaki Kawanabe. Direct importance estimation with model selection and its application to covariate shift adaptation. *Advances in Neural Information Processing Systems 20 - Proceedings of the 2007 Conference*, pages 1–8, 2009.

[228] Sugathevan Suranthiran and Suhada Jayasuriya. Signal conditioning with memory-less nonlinear sensors. *J. Dyn. Sys., Meas., Control*, 126(2):284–293, 2004.

[229] N. K. Suryadevara, S. C. Mukhopadhyay, R. Wang, and R. K. Rayudu. Forecasting the behavior of an elderly using wireless sensors data in a smart home. *Engineering Applications of Artificial Intelligence*, 26(10):2641–2652, 2013. ISSN 09521976. doi: 10.1016/j.engappai.2013.08.004.

[230] Jaakko Suutala and Juha Röning. Methods for person identification on a pressure-sensitive floor: Experiments with multiple classifiers and reject option. *Information Fusion*, 9(1):21–40, 2008. ISSN 15662535. doi: 10.1016/j.inffus.2006.11.003.

[231] B. Thélot, G. Pédrono, and L. Lasbeur. Epidemiological surveillance of falls in the elderly in france. *Revue d'Épidémiologie et de Santé Publique*, 66: S336, 2018. URL https://www.sciencedirect.com/science/article/pii/S0398762018309635.

[232] Fok Hing Chi Tivive, Abdesselam Bouzerdoum, and Moeness G Amin. A human gait classification method based on radar doppler spectrograms. 2010(1), 2010. doi: 10.1155/2010/389716.

[233] B. Ugur Toreyin, E. Birey Soyer, Ibrahim Onaran, and E. Enis Cetin. Falling person detection using multisensor signal processing. *Eurasip Journal on Advances in Signal Processing*, 2008, 2008. ISSN 16876172. doi: 10.1155/2008/149304.

[234] Kirill Trapeznikov and Venkatesh Saligrama. Supervised Sequential Classification Under Budget Constraints. *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*, 31:581–589, 2013. ISSN 15337928.

[235] Masamitsu Tsuchiya, Yuji Yamauchi, Takayoshi Yamashita, and Hironobu Fujiyoshi. Transfer forest based on covariate shift. *Proceedings - 3rd IAPR Asian Conference on Pattern Recognition, ACPR 2015*, pages 760–764, 2016. doi: 10.1109/ACPR.2015.7486605.

[236] Huan Wen Tzeng, Mei Yung Chen, and Jai Yu Chen. Design of fall detection system with floor pressure and infrared image. *2010 International Conference on System Science and Engineering, ICSSE 2010*, pages 131–135, 2010. doi: 10.1109/ICSSE.2010.5551751.

[237] Pranesh Vallabh and Reza Malekian. Fall detection monitoring systems: a comprehensive review. *Journal of Ambient Intelligence and Humanized Computing*, 9(6):1809–1833, 2018. ISSN 18685145. doi: 10.1007/s12652-017-0592-3. URL http://dx.doi.org/10.1007/s12652-017-0592-3.

[238] J Van Hulse, T M Khoshgoftaar, and A Napolitano. Experimental perspectives on learning from imbalanced data. *Proc. of the 24th International Conference on Machine learning*, pages 935–942, 2007.

[239] Tim van Kasteren, Athanasios Noulas, Gwenn Englebienne, and Ben Kröse. Accurate activity recognition in a home setting. pages 1–9, 2008. doi: 10.1145/1409635.1409637.

[240] Tom Veniat and Ludovic Denoyer. Learning Time/Memory-Efficient Deep Architectures with Budgeted Super Networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018. URL http://arxiv.org/abs/1706.00046.

[241] Ruben Vera-Rodriguez, John S.D. Mason, Julian Fierrez, and Javier Ortega-Garcia. Comparative analysis and fusion of spatiotemporal information for footstep recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(4):823–834, 2013. ISSN 01628828. doi: 10.1109/TPAMI.2012.164.

[242] Michel Verleysen, Fabrice Rossi, and Damien François. Advances in feature selection with mutual information. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5400:52–69, 2009. ISSN 03029743. doi: 10.1007/978-3-642-01805-3_4.

[243] Kevin Viard, Maria Pia Fanti, Gregory Faraut, and Jean Jacques Lesage. An Event-Based Approach for Discovering Activities of Daily Living by Hidden Markov Models. *Proceedings - 2016 15th International Conference on Ubiquitous Computing and Communications and 2016 8th International Symposium on Cyberspace and Security, IUCC-CSS 2016*, pages 85–92, 2017. doi: 10.1109/IUCC-CSS.2016.020.

[244] Gilles Virone. Assessing everyday life behavioral rhythms for the older generation. *Pervasive and Mobile Computing*, 5(5):606–622, 2009. ISSN 15741192. doi: 10.1016/j.pmcj.2009.06.008. URL http://dx.doi.org/10.1016/j.pmcj.2009.06.008.

[245] Changsheng Wan, Li Wang, and Vir V. Phoha. A survey on gait recognition. *ACM Computing Surveys*, 51(5), 2018. ISSN 15577341. doi: 10.1145/3230633.

[246] Jianwu Wang, Zhichuan Huang, Wenbin Zhang, Ankita Patil, Ketan Patil, Ting Zhu, Eric J. Shiroma, Mitchell A. Schepps, and Tamara B. Harris. Wearable sensor based human posture recognition. *Proceedings - 2016 IEEE International Conference on Big Data, Big Data 2016*, pages 3432–3438, 2016. doi: 10.1109/BigData.2016.7841004.

[247] Joseph Wang, Kirill Trapeznikov, and Venkatesh Saligrama. An LP for sequential learning under budgets. *Journal of Machine Learning Research*, 33:987–995, 2014. ISSN 15337928.

[248] Joseph Wang, Kirill Trapeznikov, and Venkatesh Saligrama. Efficient Learning by Directed Acyclic Graph For Resource Constrained Prediction. *Advances in Neural Information Processing Systems 28*, pages 2152–2160, 2015. URL http://papers.nips.cc/paper/5982-efficient-learning-by-directed-acyclic-graph-for-resource-constrained-predi pdf.

[249] Wei Wang, Alex X. Liu, Muhammad Shahzad, Kang Ling, and Sanglu Lu. Understanding and modeling of WiFi signal based human activity recognition. In *Proceedings of the Annual International Conference on Mobile Computing and Networking, MOBICOM*, volume 2015-Septe, pages 65–76, 2015. ISBN 9781450336192. doi: 10.1145/2789168.2790093.

[250] Wei Wang, Alex X Liu, and Muhammad Shahzad. Gait Recognition Using WiFi Signals. *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 363–373, 2016.

[251] Xuezhi Wang, Tzu Kuo Huang, and Jeff Schneider. Active transfer learning under model shift. *31st International Conference on Machine Learning, ICML 2014*, 4: 3102–3111, 2014.

[252] Yuxi Wang, Kaishun Wu, and Lionel M Ni. Wifall: Device-free fall detection by wireless networks. *IEEE Transactions on Mobile Computing*, 16(2):581–594, 2017. doi: 10.1109/TMC.2016.2557792.

[253] Zengmao Wang, Bo Du, Lefei Zhang, Liangpei Zhang, Ruimin Hu, and Dacheng Tao. On gleaning knowledge from multiple domains for active learning. *IJCAI International Joint Conference on Artificial Intelligence*, pages 3013–3019, 2017. ISSN 10450823.

[254] Abraham Itzhak Weinberg and Mark Last. Interpretable decision-tree induction in a big data parallel framework. *International Journal of Applied Mathematics and Computer Science*, 27(4):737–748, 2017. ISSN 20838492. doi: 10.1515/amcs-2017-0051.

[255] Abraham Itzhak Weinberg and Mark Last. Selecting a representative decision tree from an ensemble of decision-tree models for fast big data classification. *Journal of Big Data*, 6(1), 2019. ISSN 21961115. doi: 10.1186/s40537-019-0186-3. URL https://doi.org/10.1186/s40537-019-0186-3.

[256] G. Weiss. Mining with rarity: A unifying framework. *ACM SIGKDD Explorations Newsletter 6.1 (2004) 6(1) 7-19*, 6(1):262, 2005.

[257] Karl Weiss, Taghi M. Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big Data*, 3(1):9, May 2016.

[258] Karl R Weiss and Taghi M Khoshgoftaar. Investigating transfer learners for robustness to domain class imbalance. In *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 207–213. IEEE, 2016.

[259] Karl R Weiss and Taghi M Khoshgoftaar. Comparing transfer learning and traditional learning under domain class imbalance. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 337–343. IEEE, 2017.

[260] Karl R. Weiss, Taghi M. Khoshgoftaar, and Oneeb Rehman. Designing a Testing Framework for Transfer Learning Algorithms (Application Paper). *2016 IEEE 17th International Conference on Information Reuse and Integration (IRI)*, pages 152–159, 2016.

[261] Tao Xu, Yun Zhou, and Jing Zhu. New advances and challenges of fall detection systems: A survey. *Applied Sciences (Switzerland)*, 8(3), 2018. ISSN 20763417. doi: 10.3390/app8030418.

[262] Zhixiang Xu, Matt J. Kusner, Kilian Q. Weinberger, and Minmin Chen. Cost-Sensitive Tree of Classifiers. 28:133–141, 2012.

[263] Zhixiang Eddie Xu, Kilian Q. Weinberger, and Olivier Chapelle. The Greedy Miser: Learning under Test-time Budgets. *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 1175–1182, 2012.

[264] Zhixiang Eddie Xu, Matt J. Kusner, Kilian Q. Weinberger, and Alice X. Zheng. Gradient Regularized Budgeted Boosting. *arXiv preprint arXiv:1901.04065*, 2019. URL http://arxiv.org/abs/1901.04065.

[265] Bing Xue, Mengjie Zhang, Will N. Browne, and Xin Yao. A Survey on Evolutionary Computation Approaches to Feature Selection. *IEEE Transactions on Evolutionary Computation*, 20(4):606–626, 2016. ISSN 1089778X. doi: 10.1109/TEVC.2015.2504420.

[266] Yuu Yamada, Einoshin Suzuki, Hideto Yokoi, and Katsuhiko Takabayashi. Decision-tree induction from time-series data based on a standard-example split test. *In Proceedings of the 20th International Conference on Machine Learning (ICML03)*, pages 840–847, 2003.

[267] Jiashi Yang. *The Mechanics of Piezoelectric Structures*. 2010. ISBN 9812567011. doi: 10.1142/9789812774057.

[268] Jihoon Yang and Vasant Honavar. Feature Subset Selection Using A Genetic Algorithm. *Feature Extraction, Construction and Selection. The Springer International Series in Engineering and Computer Science.*, 453:117–136, 1997. doi: 10.1017/CBO9781107415324.004. URL https://doi.org/10.1007/978-1-4615-5725-8{_}8.

[269] Jun Yang, Rong Yan, and Alexander G Hauptmann. Cross-domain video concept detection using adaptive svms. In *Proceedings of the 15th ACM international conference on Multimedia*, pages 188–197. ACM, 2007.

[270] Yang Yang, Chunping Hou, Yue Lang, Dai Guan, Danyang Huang, and Jinchen Xu. Open-set human activity recognition based on micro-Doppler signatures. *Pattern Recognition*, 85:60–69, 2019. ISSN 00313203. doi: 10.1016/j.patcog.2018.07.030. URL https://doi.org/10.1016/j.patcog.2018.07.030.

[271] Lina Yao, Quan Z. Sheng, Wenjie Ruan, Tao Gu, Xue Li, Nick Falkner, and Zhi Yang. RF-Care: Device-Free Posture Recognition for Elderly People Using A Passive RFID Tag Array. (October), 2015. doi: 10.4108/eai.22-7-2015.2260064.

[272] Delaram Yazdansepas, Anzah H. Niazi, Jennifer L. Gay, Frederick W. Maier, Lakshmish Ramaswamy, Khaled Rasheed, and Matthew P. Buman. A Multi-featured Approach for Wearable Sensor-Based Human Activity Recognition. *Proceedings - 2016 IEEE International Conference on Healthcare Informatics, ICHI 2016*, pages 423–431, 2016. doi: 10.1109/ICHI.2016.81.

[273] Miao Yu, Adel Rhuma, Syed Mohsen Naqvi, Liang Wang, and Jonathon Chambers. A posture recognition-based fall detection system for monitoring an elderly person in a smart home environment. *IEEE Transactions on Information Technology in Biomedicine*, 16(6):1274–1286, 2012. ISSN 10897771. doi: 10.1109/TITB.2012.2214786.

[274] Mitchell Yuwono, Steven W. Su, Bruce D. Moulton, and Hung T. Nguyen. Gait cycle spectrogram analysis using a torso-attached inertial sensor. *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*, (1):6539–6542, 2012. ISSN 1557170X. doi: 10.1109/EMBC.2012.6347492.

[275] Hans Zantema. Decision Trees: Equivalence and Propositional Operations. 1998.

[276] Hans Zantema and Hans Bodlaender. Finding small equivalent decision trees is hard. *International Journal of Foundations of Computer Science*, 1999.

[277] Chao Zhang, Lei Zhang, and Jieping Ye. Generalization bounds for domain adaptation. *Advances in Neural Information Processing Systems*, 4:3320–3328, 2012. ISSN 10495258.

[278] Chenyang Zhang and Yingli Tian. RGB-D Camera-based Daily Living Activity Recognition. *Journal of Computer Vision and Image Processing*, 2:12, 2012. URL http://media-lab.ccny.cuny.edu/wordpress/Publications/NWPJ-201209-15-CameraReady.pdf.

[279] Kun Zhang, Bernhard Scholkopf, Krikamol Muandet, and Zhikun Wang. Domain adaptation under target and conditional shift. *30th International Conference on Machine Learning, ICML 2013*, 28(PART 3):1856–1864, 2013.

[280] Yifan Zhang, Peilin Zhao, Jiezhang Cao, Wenye Ma, Junzhou Huang, Qingyao Wu, and Mingkui Tan. Online adaptive asymmetric active learning for budgeted imbalanced data. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2768–2777, 2018. doi: 10.1145/3219819.3219948.

[281] Zhong Zhang, Christopher Conly, and Vassilis Athitsos. A survey on vision-based fall detection. *8th ACM International Conference on PErvasive Technologies Related to Assistive Environments, PETRA 2015 - Proceedings*, 2015. doi: 10.1145/2769493.2769540.

[282] Qiangfu Zhao and Mitsuyoshi Shirasaka. A study on evolutionary design of binary decision trees. *Proceedings of the 1999 Congress on Evolutionary Computation, CEC 1999*, 3(2):1988–1993, 1999. doi: 10.1109/CEC.1999.785518.

[283] Xiaodan Zhuang, Jing Huang, Gerasimos Potamianos, and Mark Hasegawa-Johnson. Acoustic fall detection using Gaussian mixture models and GMM supervectors. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, pages 69–72, 2009. ISSN 15206149. doi: 10.1109/ICASSP.2009.4959522.

[284] Nadia Zouba, Bernard Boulay, Francois Bremond, and Monique Thonnat. Monitoring activities of daily living (ADLs) of elderly based on 3D key human postures. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5329 LNCS:37–50, 2008. ISSN 03029743. doi: 10.1007/978-3-540-92781-5_4.

université PARIS-SACLAY | ÉCOLE DOCTORALE de mathématiques Hadamard (EDMH)

Titre : Monitoring de personnes âgées basé sur des modèles d'arbres de décision en conditions de variation de domaines et de ressources computationnelles limitées.

Mots Clefs : Monitoring de personnes âgées, forêts aléatoires, apprentissage par transfert, systèmes embarqués, contraintes computationnelles.

Résumé : Cette thèse porte sur le monitoring de personnes âgées en maisons de retraite et EHPAD, à partir d'un capteur sol piezo-électrique relié à un système embarqué. Après avoir présenté les particularités du signal piezo-électrique ainsi que sa conversion en séries temporelles, nous expliquons comment les différentes technologies de mesures affectent le signal original et présentons les différents environnements de collecte de données. Pour notre application un vaste ensemble de features est employé, basé sur plusieurs représentations du signal comme la transformée de Fourier, l'auto-corrélation et les spectrogrammes. Plusieurs études expérimentales sur les forêts aléatoires sont réalisées et montrent la faisabilité de plusieurs tâches de détection ainsi que la pertinence des différentes représentation du signal et features associées.
Cependant pour être déployés de manière industrielle ces modèles doivent respecter deux contraintes majeures. Premièrement ils doivent être confrontés aux données réelles et s'adapter à la variabilité des installations et des patients. Dans cette optique des méthodes d'apprentissage par transfert sont étudiées pour l'intégration de nouvelles données obtenues en conditions réelles dans un modèle préalablement entraîné sur des données simulées en environnement contrôlé. Nous étudions donc les effets néfastes liés au déséquilibre de classes sur l'apprentissage par transfert et proposons, pour palier à ce problème, des adaptations de méthodes existantes sur les arbres de décision. A partir de ces adaptations nous développons un algorithme robuste de transfert sur les forêts aléatoires utile à la fois pour gérer le déséquilibre de classes et pour interpréter les relations entre les différents domaines de données.
En outre ces modèles doivent tourner en temps réel dans un système embarqué aux capacités computationnelles réduites. Prendre en compte ce genre de contraintes lors de la conception d'un modèle de prédiction correspond au domaine de recherche appelé "budgeted learning", qui est particulièrement actif ces dernières années. Nous définissons formellement le temps de prédiction pour les arbres de décision et forêts aléatoires prenant en compte à la fois le coût d'acquisition des features et le coût d'évaluation du modèle. Nous proposons un algorithme génétique pour résoudre le problème de la contrainte du temps de prédiction qui permet de passer d'un modèle déjà entraîné sans contraintes à un modèle simplifié qui respecte cette contrainte de temps de calcul. Cet algorithme tire parti d'un pruning aléatoire et de la notion d'équivalence entre les arbres de décision, c'est-à-dire lorsque deux modèles représentent la même fonction de décision mais diffèrent par leur structure, dans le but de favoriser la réduction du temps de prédiction en exploitant la variété structurelle des arbres de décision.

Title : Elderly monitoring using decision trees under domain shifts and computational resource constraints.

Keys words : Elderly monitoring, random forests, embedded systems, transfer learning, budgeted learning.

Abstract : Tarkett is a global flooring company that developed a piezo-electric sensor encapsulated in the flooring and an embedded system meant to be equipped in nursing home patient rooms. The objective through this industrial project is to build reliable machine learning models able to work in real-time in the embedded system, based on piezo-electric signals, to provide useful information for medical staff to monitor their patients health.
Considering different measurement technologies we describe how they affect the original physical signal, as well as different data gathering environments in which several dataset have been recorded. To be able to monitor elderly health state some important recurrent events like walk and some anomalies like falls need to be recognized from floor sensor signals. To this end, the way to process signals into adequate data representation, according to these detection purpose, is also a major challenge. We use a wide feature set based on time series from various signal representations such as Fourier transform, autocorrelation and spectrograms. Using predictive models based on random forests on different experimental datasets we show Tarkett system ability to achieve various monitoring tasks, as well as the relevance of each signal representation and associated features regarding these detection tasks.
Nevertheless for these experimental studies to be deployed industrially in FIM Care real installations, machine learning models need to fulfill two crucial requirements. Firstly they have to be confronted with real environment data, meaning to be able to adapt to real installations variability and to activity signal differences between people. In this context we deal with the problem of adapting a predictive model initially trained on experimental data to real data with different empirical distribution. This particular situation in machine learning is known as *transfer learning* or *domain adaptation*. We address it by confronting simulated events data to real data on the *fall detection* task that presents the particularity of extreme *class imbalance* in real conditions. We investigate the drawbacks of this *class imbalance* on existing *transfer learning* methods on decision trees and propose some adaptations to handle this problem. Our contribution is a robust model-based *transfer learning* algorithm on random forests able to deal with *class imbalance* and that can also be used to interpret relations between two different domains.
Secondly, most of the prediction tasks for elderly monitoring have to work in real time being embedded in an electronic device with limited computational capabilities. Taking into account this kind of constraints while designing a predictive model belongs to a branch of machine learning, known as *cost sensitive* or *budget learning*, that became an increasingly active research topic in the past years. We translate embedded system computational resource constraints into a *budgeted prediction time* framework compatible with decision tree based models and propose an efficient and scalable genetic algorithm considering both *feature acquisition cost* and *evaluation cost* allowing to pass from an experimental random forest model to a new simplified one that fits in embedded system resource limits. This algorithm takes advantage of the notion of *equivalence* between classifiers, meaning models sharing the same decision function but with different structures, to favor *feature acquisition cost* reduction by exploiting structural variety on decision trees.