Personalized and Weakly Supervised Learning for Parkinson's Disease Symptom Detection

Ada J. Zhang

CMU-RI-TR-20-01

October 21, 2019

Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

This material is based upon work supported by the CMU Center for Machine Learning and Health Fellowship, the National Science Foundation under Grant No. IIS-1602337, and the National Science Foundation Graduate Research Fellowship under Grant No. DGE-1252522. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Keywords: Personalization, human motion monitoring, in-the-wild, Parkinson's disease, tremor detection, transfer learning, domain adaptation, importance re-weighting, Support Vector Machine (SVM), weakly supervised learning, multiple instance learning, stratified multiple instance learning.

Abstract

Parkinson's Disease (PD) is a neurodegenerative disorder that affects approximately one million Americans. Medications exist to manage the symptoms, but doctors must periodically adjust dosage level and frequency as a patient's disease progresses. These adjustments are typically based on observations made during short clinic visits, which provide an incomplete picture of a patient's daily quality of life. To provide a more accurate assessment of a patient's at-home experience, this thesis explores the use of wearable accelerometers for monitoring PD in the wild. A central challenge of this application is human variability, which makes it difficult for algorithms to generalize to "unseen" subjects who were not in the training set. This thesis investigates two major approaches to address this issue: (1) stratified algorithms for learning from in-the-wild data, and (2) personalization algorithms to actively adjust for human variability.

A major challenge with in-the-wild data collection is obtaining labels of tremor symptoms. We developed a data collection protocol in which subjects used a cellphone app to log the *approximate* ([0-33%], [33-66%], or [66-100%]) amount of tremor experienced within short time segments and collected in-the-wild data from six PD patients. We call these approximate percentages *stratified* labels, and we present a novel stratified weakly supervised learning technique that benefits from labels with this structure. Experiments demonstrated that this new technique leads to higher performance on this dataset compared to standard weakly supervised learning algorithms. Furthermore, stratified weakly supervised algorithms performed better on in-the-wild data than fully supervised algorithms trained on accurately labeled laboratory data. These findings suggest that collecting stratified labels from each subject in the wild is more effective than collecting accurate labels from other subjects in the laboratory. In summary, stratified learning provides a practical and efficient method for collecting weak labels in-the-wild that greatly improves accuracy of person-specific PD tremor detection.

This thesis also explores several unsupervised personalization algorithms, which either reweight the training data to align its distribution with the test data or learn a regression from distributions to classifiers. We extended these personalization algorithms to the multiclass scenario. In our dataset, the personalization algorithms demonstrated modest improvement over non-personalized algorithms. We further analyzed the behavior of these algorithms with synthetic data, gaining insights into their limitations when learning appropriate weights. We addressed this issue by developing new constraints to the cost function, which improved performance on synthetic data, and experimented with global optimization.

Acknowledgments

I must first thank my advisors, Prof. Fernando De la Torre and Prof. Jessica Hodgins, for guiding me through this long journey. Objective functions of machine learning algorithms used to seem like inscrutable, magical, black box inventions. Fernando taught me to think of them as mathematical descriptions of desired solutions, empowering me to make calculated modifications to the objective functions and tailor them to the problem at hand. Fernando also helped curb the perfectionist in me, often reminding me to "be practical." Meanwhile, Jessica held us to high standards, always making sure we were asking the right questions and investigating them thoroughly. I also looked to Jessica as a role model for women in computer science, and I particularly admired her confidence: she is completely secure in her strengths, but what's more impressive is her ability to admit her weaknesses. I hope someday to have Fernando's machine learning intuition and practicality, and Jessica's high standards and confidence.

I would also like to thank Prof. Artur Dubrawski and Dr. Anthony (Tony) Jarc for serving on my committee. Artur provided valuable insights on machine learning and encouraged me when I felt like nothing was working. Tony gave me the opportunity to intern at Intuitive Surgical, Inc., a company I had long admired, and made sure I addressed the practical applications of my work.

Throughout the years, many different lab members have helped me in many ways, including brainstorming, moral support, and setting up computers/servers. Several were vital to the development of the dataset presented in this thesis: Dr. Andrew Whitford, Alexander Cebulla, and Dr. Stanislav Panev helped collect the data, Vicent Ortiz and Alexander Cebulla implemented the cell phone app, Katelyn Stebbins labeled the data, and Dr. Robyn Massa supplied UPDRS scores.

Many friends supported me – too many to name; but three deserve special recognition. Vincent Chu played a large role in my first paper with Fernando, helping me design the experiments and the story. Working with Vincent gave me the momentum I needed to start moving forward and construct my thesis proposal. Nate Brooks was my first "Pomodoro Buddy." Every day, we motivated each other to sit down, press "start," and work. Five months of pomodoros later, I successfully proposed my thesis. After Nate defended, I was so grateful to meet Ivan Ruchkin, who stepped in as Pomodoro Buddy #2. Ivan provided encouragement when everything seemed to be failing and helped me identify the cognitive distortions present when telling myself that I was a "bad grad student." Answering his almost daily questions about machine learning boosted my confidence, showing me how much I had learned in my time at CMU. I thank Vincent, Nate, and Ivan for getting me through some of the hardest times of my Ph.D.

Last, but certainly not least, I want to thank my family. In particular, I thank Christian Marcheselli for standing with me through all the ups and the downs, and for waiting so patiently these eight and a half years.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Chronic neurogdegenerative disorders like Parkinson's disease (PD) pose a serious threat to the elderly population. As many as one million Americans (mostly aged 65 or older) live with PD [95], which is more than the combined number of people diagnosed with multiple sclerosis, muscular dystrophy and Lou Gehrig's disease. The combined direct and indirect cost of PD, including treatment, social security payments and lost income from inability to work, is estimated to be nearly $25 billion per year in the United States alone [95]. The number of PD patients and the cost associated with them can be expected to grow as our population ages.

People who suffer from PD experience a variety of motor and non-motor symptoms. Perhaps the most well-known symptom of PD is tremor. However, PD causes additional motor symptoms such as bradykinesia (slowness of movement), stiffness, and rigidity. These often result in postural instability, gait problems (which can lead to increased frequency of falling), and reduced facial expressions. PD also causes several non-motor symptoms, including cognitive impairment, mood disorders (depression and anxiety), sleep problems, speech/swallowing problems, and unexplained pain [89]. The combination of motor and non-motor symptoms can reduce a patient's ability to remain self-sufficient, affect relationships with friends and family, and in general, result in a lower quality of life.

There is currently no cure for PD and existing medication can only provide symptomatic relief. Sensitivity to these drugs decreases over time; patients require larger and more frequent doses. Furthermore, some medications may cause severe side effects, such as dyskinesia, which manifests as an involuntary, jerky movement. Therefore, doctors work to find the minimal dosage that will manage their patients' symptoms. In this way, they hope to prolong the efficacy of the medication and postpone the onset of side effects. The current standard of care is as follows: a patient meets with his or her doctor every three to six months, self-reporting on symptoms and response to medication. The doctor then performs a motor function assessment by examining the patient's performance on motor tasks selected from Part III of the Unified Parkinson's Disease Rating Scale (UPDRS) [42]. Finally, the doctor adjusts medication dosage as necessary.

There are several shortcomings to the current state-of-the-art in PD management:

- Frequent clinic visits are a major contributor to the high cost of PD treatment and are inconvenient for the patient, especially in a population for which traveling may be difficult.

- Inaccurate patient self-reports and 15-20 minute clinic visits do not provide enough information for doctors to accurately assess their patients, leading to difficulties in monitoring patient symptoms and medication response.
- Motor function assessments are not only subjective, but also dependent on the time of day and time since the last medication intake. All of these factors make it difficult to monitor disease progression via short and relatively infrequent office visits.

This thesis aims to explore a new paradigm shift in PD management through 24/7, in-home monitoring using low-cost sensors and machine learning. Specifically, it explores the use of wearable accelerometers to detect and quantitatively assess the severity of PD motor symptoms, tremor in particular, during daily living activities in the home environment. Such a system would result in several positive impacts: (1) clinicians would be better able to track their patients' symptom occurrences, medication response and disease progression, (2) improved monitoring would enable a more personalized drug-therapy regimen, (3) continuous monitoring of motor function could allow for less frequent clinic visits, and (4) quantitative assessment of motor symptoms could be developed into an objective measurement of PD severity, which could facilitate research of novel drugs or neurotherapies, the effects of which are currently too subtle to detect. In short, such a system would result in greater convenience to the patient, reduced cost to society and, most importantly, better care.

Using machine learning for in-home detection of PD symptoms from accelerometer signals is challenging for two reasons:

- **Generalization to unseen patients and personalization to specific patients:** Our long-term goal is to build a system that can be distributed to every PD patient. Because we can only train on a finite, sample population, the machine learning algorithms must be able to generalize to "unseen" people who were not in the training population. Different people are affected by the progression of PD in different ways, which implies that accelerometer data will differ greatly across subjects. Studies have found that algorithms trained on data from specific people perform better on those people than algorithms trained on a cohort of other people [25, 132]. However, such algorithms require labeled data from each new user, which, for certain applications, may not be available. It is a challenge, therefore, to improve the performance of a generic classifier by personalizing it to a specific person in an *unsupervised* manner, *i.e.*, without having access to new labels.
- **Generalization from the lab to the wild:** Previous work on PD symptom detection (see Table 2.4) has focused on data collected from controlled environments, where participants are asked to perform a prescribed set of tasks in a particular order and where the accelerometer data can be accurately labeled by referencing additional video data. This protocol, with accurate labels of the start and end of each symptom, allows researchers to use standard, supervised learning techniques. The assumption is that models trained on such data will maintain their performance when applied "*in*

the wild": *i.e.*, in patient homes and during routine activities of daily living. It is unclear if this assumption truly holds: laboratory data has lower variability in the types of activities performed and participants may move differently in the lab versus in their own homes. However, to address this assumption, accelerometer data must be collected in the wild, where accurate labels of the exact start and end of each symptom are difficult to obtain. Approximate labels of symptom occurrences are more feasible, but algorithms need to accommodate this additional uncertainty.

This thesis directly addresses these two challenges.

In Chapter 3, I describe a protocol I designed for laboratory and in-the-wild data collection. The laboratory data collection involved motor tests from the Unified Parkinson's Disease Rating Scale (UPDRS) and some activities of daily living, such as making a sandwich, eating, playing cards, writing a letter, etc. These activities were recorded with three cameras to minimize occlusion, and the start and end of every symptom occurrence was labeled by referencing the video data. To allow participants to label their data in the wild, we built a cellphone app that prompted users approximately every hour to describe the symptoms they had experienced in the previous five minutes. In particular, they were asked to select from three options – "Almost none," "Half the time," and "Almost always" – to describe their tremor symptoms in the past five minutes. To prevent users from backdating entries, the app would not allow users to submit labels more frequently than once per hour, and it automatically collected time stamps. Users were paid per label to encourage more frequent labeling.

Several algorithms exist for personalizing classifiers to new subjects given their unlabeled data. However, these algorithms have not been applied to PD motor symptom detection. In Chapter 4, I describe three of these algorithms – Kernel Mean Matching (KMM), Selective Transfer Machine (STM), and Transductive Parameter Transfer (TPT) – and compare them to a generic Support Vector Machine (SVM). I find that the personalization algorithms do show some improvement over the generic classifier, especially when optimal parameters are chosen. However, finding the optimal parameters automatically is difficult and time-consuming. Results show that STM, when given the best-performing parameters, outperforms other algorithms with their respective best-performing parameters. That is, STM has higher *potential* for performance improvement. However, in real scenarios, the parameters that perform best on test data are unknown. Instead, they must be chosen using cross validation on training data. Under this paradigm, experiments show that KMM leads to greater improvement.

Chapter 5 presents in-depth experiments on STM to better understand the behavior from Chapter 4. In particular, STM is applied to a human activity recognition dataset and several synthetic datasets. Because these datasets are multiclass, this chapter develops a multiclass formulation of STM. Based on the failure modes of STM found by examining its behavior on synthetic data, two improvements to the STM objective function are proposed. These modifications demonstrate good performance on the synthetic data, but have more modest performance on the activity recognition dataset. These findings lead to an analysis of local stationary points and global optimization of STM, the results of which imply that the objective function of STM is not monotonically related to accuracy on these datasets.

In Chapter 6, I compare several weakly supervised learning algorithms, which are designed to learn from *weak labels*. In contrast to accurate labels, which indicate the exact start and end of every event, standard weak labels only indicate whether or not events occurred within a specific window of time. I develop an alternative method of weakly labeling data, which I call stratified labeling, and which gives more nuanced information than standard weak labels. I also develop a modification to the standard weakly supervised algorithms that allows them to process these stratified weak labels. All algorithms are compared to a naive baseline, which represents the simplest method of applying a fully supervised algorithm to weakly labeled data. Performance is assessed on the laboratory data, where accurate labels could be used to compute weak labels on time segments of varying lengths. In this way, the change in algorithm performance as time segments increased in length could be analyzed. Results show that the stratified algorithms are more successful at sustaining performance when segment length increases compared with the non-stratified algorithms.

In Chapter 7, I analyze the legitimacy of the assumption that training on laboratory data will generalize to in-the-wild data. It is hypothesized that laboratory data differs from in-the-wild data not only due to differences in activity variability, but also because weak labels in Chapter 6 were computed from accurate labels, whereas weak labels in wild data could be subject to varying interpretations by the patients. To test this hypothesis, three models are trained: (1) a generic (fully supervised) SVM trained on laboratory data of all subjects excluding the test subject, which represents the standard leave-one-subject-out paradigm; (2) a person-specific (fully supervised) SVM trained on laboratory data of the test subject only, and which has previously been shown to have better performance than the generic SVM; and (3) a person-specific (weakly supervised) SVM trained on in-the-wild data of the test subject. The performance of these three algorithms is compared on both laboratory and in-the-wild data. Results show that, for all three algorithms, there is a large discrepancy in performance between laboratory and in-the-wild data. These findings are consistent with the hypothesis that differences in activities and label interpretation prevent results on laboratory data from generalizing to data in the wild. These differences are not present when training and testing on wild data. However, it is interesting that, when testing on wild data, training on the less precise labels from wild data outperforms training on accurate labels from laboratory data.

In summary, this thesis explores personalization and weakly supervised learning for improved PD symptom detection in-the-wild. The main contributions of this thesis are as follows:

- **Collection of an in-home, PD tremor dataset using a cellphone application for labeling tremor symptoms.** The app was designed to promote more regular and frequent submission of labels in addition to higher temporal accuracy compared to traditional paper diaries. A link to this dataset can be found at `http://www.humansensing.cs.cmu.edu/software`.
- **Development of a novel form of weak labels, called "stratified" labels, and associated stratified weakly supervised algorithms.** Traditional weak labels only indicate whether or not an event of interest occurs during a segment of time. These

labels are useful when such events occur in relatively short time-spans. In contrast, PD symptoms are often more continuous in nature and could encompass none of the time-span, all of the time-span, or any percentage between. Stratified labels indicate the approximate percentage of PD symptoms within a time segment. They are therefore more nuanced and more appropriate than standard weak labels for this application. The stratified algorithms developed to process these stratified labels were shown to have higher performance than their standard counterparts and were able to maintain their performance even as the length of the time segments increased.

- **In-depth analysis of STM on synthetic data and accelerometer data from a human activity recognition dataset, and development of several extensions to the STM algorithm.** Because human activity recognition is a multiclass problem, a new objective function was formulated such that STM could solve for multiple classes *simultaneously*. Furthermore, based on the behavior of this STM in synthetic data, two modifications to the constraints of the objective function were developed. One modification restricted the total weight assigned to each class to be equal. The other modification restricted the total weight assigned to be proportional to the class distribution. These modifications were shown to have higher performance than the original STM in synthetic data. Additional analyses indicated that, while STM has a local stationary point at a solution with high accuracy on the test set, the STM objective function is not monotonically related to accuracy.
- **Analysis and assessment of several personalization algorithms – KMM, STM, and TPT – applied to PD data and compared to the generic SVM counterpart.** Interestingly, these personalization algorithms showed more modest improvement over the generic algorithm on this dataset than what was previously reported on other datasets. These findings led to the more in-depth analyses described above.
- **Assessment of the validity of the common assumption that results on laboratory data will transfer to the wild.** Algorithms were trained on both LAB and WILD data, and then tested on LAB and WILD data. All algorithms exhibited large changes in performance between LAB and WILD data, indicating that the two datasets differ substantially. Interestingly, training on less precise labels from WILD data led to better performance on WILD data than training on accurate labels from LAB data. These findings suggest that LAB data are a poor representation of WILD data, and that it is more beneficial to collect less precise labels in the wild than accurate labels from the laboratory.

While this thesis focuses on PD tremor detection, the findings and suggestions should be applicable to other PD symptoms, other motor control diseases, and human motion understanding in general. These contributions are detailed in Chapter 8, where I also discuss avenues for future research.

Table 1.1 lists some of the notation that is consistent throughout the thesis. Superscripts and subscripts may slightly modify the meaning of the symbols. In these cases, the exact term will be defined immediately before or after its usage. Notation specific to a particular algorithm or objective function is defined when those concepts are introduced.

Table 1.1: Mathematical notation

Symbol	**Meaning**
N	Number of training subjects
n_{tr}	Number of training samples
n_{te}	Number of test samples
N_{seg}	Number of time segments
d	Number of features
m	Number of classes
$\mathbf{x}_i$	A $d \times 1$ feature vector
y_i	Label of the ith training point, $y_i \in \{+1, -1\}$
$\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{n}$	A set of training samples.

Chapter 2

Related Work

This thesis applies techniques from human activity recognition and personalization to the problem of PD symptom detection. It also extends the current work in PD symptom detection to directly address detection in the wild. The following sections summarize relevant related work in human activity recognition, personalization (*i.e.*, machine learning under covariate shift), and applications of machine learning and wearable sensors to automated PD assessment.

2.1 Human Activity Recognition

Human activity recognition (using video, accelerometers, or motion capture) has been a well-studied field due to its host of possible applications, including surveillance, human computer interaction, patient monitoring, exercise assessment and fitness tracking. Much of the early work in this field focused on activity recognition from video data (see [2] and [69] for a review). Similar to video data, some work has focused on using 3D data, either from stereo cameras, motion capture systems, or range sensors, like the Microsoft Kinect (see [3] for a review). More recently, researchers have moved to other sensing modalities, such as the accelerometer or GPS sensors within mobile phones (see [64] for a review).

Most relevant to the work in this thesis is that of human activity recognition through accelerometers. In general, features are computed over windows of the accelerometer signal, where the window length varies widely and is often application dependent. Numerous features have been explored, including time-domain features (such as the mean, root mean square, standard deviation, or mean absolute deviation) and frequency domain features (such as the energy in a frequency band or entropy). Many different algorithms have also been explored, including decision trees, naive Bayes, k-nearest neighbors (k-NN), hidden Markov models, conditional random fields, and SVMs. We refer the reader to [76] for a comprehensive review.

Three methods are commonly used to evaluate these features and algorithms:

- *User specific* – Algorithms are trained and tested on data from the same subject. [14, 107, 125, 142]

- *k-fold cross validation* – The training data are split randomly into k folds. The algorithm is trained on $k-1$ folds and tested on the left out fold. This process is repeated over all k folds and results across the folds are then averaged. [53, 85, 125]
- *Leave-one-subject-out (LOSO)* – The algorithm is trained on all subjects excluding one, and then tested on the left out subject. The process is repeated over all subjects, and the results across the subjects are then averaged. [14, 23, 53, 54, 55, 65, 85, 125]

The LOSO method is one of the most common evaluation methods because it is the most realistic: systems that are deployed to a broad user base will have to classify activities performed by people who were not in the training set. However, studies comparing the different methods generally find that performance worsens when evaluated with LOSO [14, 53, 70, 85, 125].

2.2 Personalization via domain adaptation

In this thesis, personalization refers to leveraging *unlabeled* data from the left-out or "test" subject so as to improve upon a classifier or model trained on labeled data from other people. This problem statement is closely related to the field of domain adaptation (also called domain transfer and transfer learning). Problems in this broad field are characterized by having differences between the source (training) and target (test) domains. For example, one may need to transfer from one feature set to another [61], or from one set of classes to another [141]. Jiang [66] and Pan and Yang [93] are two well-organized surveys describing the different types of transfer learning. Cook et al. provide a comprehensive review of transfer learning in activity recognition [28].

This thesis explores a sub-problem of domain adaptation called covariate shift, which describes the situation where the source (training) and target (test) data are sampled from different distributions. Such a situation occurs commonly in human-related applications. In the case of human activity recognition, the source domain would be composed of data from many different people encompassing many different styles of movement. In contrast, the target domain would consist of data from a single person. These data would therefore be localized around a particular style of movement rather than being sampled uniformly from all possible styles of movement. Hence, the source and target data are sampled from different distributions.

The following sections highlight two main approaches to solving the covariate shift problem and also describe work on personalization specifically for human applications.

2.2.1 Approach #1: Find an alternate feature representation in which the distributions are more aligned

Many methods for finding new feature spaces have been proposed. The "Frustratingly Easy" approach, proposed by Daumé [33], simply expands the feature space three fold, allocating one part of the new feature space to contain only features from the source domain and zeros otherwise, another part to contain only features from the target domain, and the third part to contain features from both source and target domains. (Note, however, that this approach requires some labels from the target domain, which does not fulfill the original assumption stated above of leveraging *unlabeled* data from the "test" subject.) Using the fact that the distance between two distributions can be estimated by the distance between their means in Reproducing Kernel Hilbert Space (RKHS) space, Pan et al. [94] proposed transfer component analysis (TCA), which maps the source and target domains into a subspace that minimizes the distance between their means in RKHS space. Gong et al. [43] proposed the geodesic flow kernel (GFK), which is used to derive low-dimensional representations that are invariant to the domains. Fernando et al. [39] described a subspace alignment (SA) technique, where the source and target data are projected into a subspace built from their eigenvectors, and then a transformation is applied to align the basis vectors from the two subspaces.

These approaches have generally been applied to language or visual data. On these datasets, mapping the original features into the proposed subspaces has been shown to help improve performance on the target domain. However, in medical applications, high value is placed on the *interpretability* of machine learning algorithms because it enables clinicians to understand the reasoning behind an algorithm's output and thereby make an informed final decision. Features initially computed from data are often easily understandable (maximum acceleration, dominant frequency, *e.g.*), but mapping these features into a new subspace can obfuscate their meaning. Therefore, this thesis focuses more on methods falling under the following category of approaches.

2.2.2 Approach #2: Reweight the training data to minimize the distribution mismatch.

Approaches for finding an appropriate reweighting generally fall into two categories. One method is to implicitly estimate the distributions of the training and test data, and then calculate the weights (or "importance") as a ratio of those estimates. The Kullback-Leibler Importance Estimation Procedure (KLIEP) estimates these weights by minimizing the Kullback-Leibler (KL) divergence between the test distribution and the weighted training distribution [123]. Later, a more computationally efficient estimation was proposed, called unconstrained least-squares importance fitting (uLSIF) [67]. These works are highly theoretical and were validated on standard classification and regression datasets. In a simpler method for importance estimation – nearest neighbor-based importance weighting (NNeW), proposed by Loog [83], each test sample adds one to the weight of the nearest training sample. On the same classification datasets, NNeW was found to generally outperform KLIEP and uLSIF. The same researchers that developed KLEIP and uLSIF later applied their uLSIF weights to a probabilitic classifier for distinguishing walking from running, cycling, or taking a train given accelerometer data from an iPodTouch [48]. They found that all importance reweighted algorithms performed better than their generic counterparts, and that their probabilistic classifier out performed the Laplacian regularized least-squares and kernel logistic regression algorithms.

Another strategy for estimating importance is to measure the distance between distributions and weight the training data to minimize that distance. One line of research was built from the $d_{\mathcal{A}}$-*distance* between two distributions, which intuitively measures the largest change in probability of a set of points in feature space [72]. Formally, let $\mathcal{A}$ be the set of subsets of the feature space X, and let P and Q be two distributions over X. Then, the $d_{\mathcal{A}}$-distance is given by

$$d_{\mathcal{A}}(P,Q) = \sup_{a \in \mathcal{A}} \left| P(a) - Q(a) \right| .$$

Building on the $d_{\mathcal{A}}$-distance, Mansour et al. [86] developed the *discrepancy distance*, which measures the distance between distributions given a set of classification functions. Formally, let H be a set of classification functions and let $\mathcal{L}$ be a loss function, then the discrepancy distance is given by

$$\operatorname{disc}_{\mathcal{L}}(P,Q) = \max_{h,h' \in H} \left| \mathcal{L}_P(h',h) - \mathcal{L}_Q(h',h) \right| .$$

Intuitively, if one thinks about h as the true labeling function, then the discrepancy distance measures how much the loss function value will differ when computed over the source and target distribution given a hypothetical learned classification function. The goal, therefore, is to reweight the source distribution so as to minimize the discrepancy distance. Cortes et al. [29] further extended this concept with their *generalized discrepancy distance*, which restricts the set of classification functions considered, thereby allowing the choice of weights to be less conservative. This algorithm was tested on several baseline domain adaptation datasets and found to be on par with or slightly better than other domain adaptation algorithms.

An additional method for selecting weights, proposed by Gretton et al. [45], is to estimate distribution mismatch by measuring the distance between the means of the distributions in RKHS and learn a set of weights that minimize that distance. This method is called kernel mean matching (KMM). Chu et al. [25] extended this technique by connecting it with a support vector machine (SVM) classifier and considering the loss from the current SVM hypothesis while estimating weights with KMM. This algorithm, called the Selective Transfer Machine (STM), was applied to facial action unit detection and found to improve area under the curve values by five to ten points. Another, somewhat related, technique is the transductive parameter transfer (TPT) algorithm proposed by Sangineto et al. [114]. It does not explicitly reweight the training data, but it learns a regression from data distributions to classifiers.

2.2.3 Applications of personalization in human applications

The issue of performance drop on subjects whose data were not in the training set has been well-documented and there have been many proposed solutions. The majority of these solutions, however, assume that there is a small amount of labeled data from this subject. Cvetkovic et al. [31] trained a person-specific and a generic classifier, then used a meta-classifier to predict the confidence of a label. Samples from the test subject that were labeled with high confidence were added into the training set to update the generic classifier. Results on a dataset of five subjects performing eight activities showed that the generic classifier was able to improve over time. In [124], Sun et al. jointly learned the similarity between people and a person-specific classifier for each person. When tested on 20 individuals from the ALKAN dataset, results showed that their method performed better than the generic or regular person-specific models. Lockhart and Weiss [82] built generic, person-specific, and hybrid (training set includes data from the test subject and other users) models. On a dataset of 59 users performing six activities with a smartphone, they found that the person-specific models performed best, although the hybrid models were very similar. Hong et al. [59] trained many person-specific classifiers and then used a small amount of labeled data from the test subject to build a new classifier from the set of pre-trained ones. On a dataset of 28 people wearing seven sensors and performing seven actions, they found that their personalized model was able to outperform the person-specific model. Bleser et al. [17] presented a similar approach, except that instead of building a classifier from pre-trained person specific classifiers, they applied weighted majority voting to those pre-trained classifiers, where the weights were learned using labeled data from the test subject. On a dataset of nine subjects performing 18 different activities, their confidence-based AdaBoost algorithm resulted in a 10% reduction in error rate when compared with a generic AdaBoost algorithm. Another method of personalization was presented by Rokni et al. [108], where a convolutional neural net (CNN) was trained using data from other people, and then the final, classification layer was retrained given labeled data from a new subject. The authors tested this method on two datasets and found that their model outperformed several standard machine learning algorithms, although they did not compare their personalized CNN to a generic one.

The above papers all relied on the availability of labels from the test subject. However, as mentioned above, this thesis focuses on personalization when labels are not available. This problem is considerably harder, and therefore has been less well-explored. In [140], Zhao et al. trained a decision tree using labeled data from one subject of a training population. This decision tree was then used to define cluster centers for the another subject's data. These cluster centers were then used to initialize k-means, and the decision tree was retrained. The authors found that, over iterations, the performance of the decision tree improved. However, they did not compare this personalized decision tree to algorithms trained on the entirety of the training data. Hassan et al. [52] compared several of the importance reweighting algorithms described above (KLIEP [123], uLSIF [67], and KMM [45]) to emotion recognition in acoustic data. They found that the performance of the importance reweighted were similar, offering approximately 5% improvement in accuracy compared to a generic SVM. Barbosa et al. [15] compared domain adaptation techniques, including KMM [45], TCA [94], NNeW [83], and SA [39], on two human activity recognition datasets. KMM, NNeW, and SA were generally able to improve upon the generic classifier accuracy by 5-10%. Meanwhile, TCA led to worse accuracy than the generic classifier, which they hypothesized was due to only having a single source dataset, making it more difficult to discover transferable knowledge between the source and target datasets.

Table 2.1: MDS-UPDRS Part III, Motor examination, adapted from Goetz et al. [42]

Item	Description
3.1 Speech	Volume, modulation, clarity, including slurring, palilalia (repetition of syllables) and tachyphemia (rapid speech, running syllables together) are assessed.
3.2 Facial expression	Eye-blink frequency, masked facies or loss of facial expression, spontaneous smiling and parting of lips are assessed.
3.3 Rigidity	Sit in a relaxed position while examiner manipulates limbs and neck to assess slow passive movement of major joints.
3.4 Finger tapping	Tap index finger on the thumb 10 times as quickly and as big as possible.
3.5 Hand movements	Make a tight fist and open the hand 10 times as fully and as quickly as possible.
3.6 Pronation/supination of hands	Extend arm in front and turn palm up and down alternately 10 times as fast and as fully as possible
3.7 Toe tapping	While sitting, place heel on the ground and tap the toes 10 times as big and as fast as possible.
3.8 Leg agility	While sitting, raise and stomp the foot on the ground 10 times as high and as fast as possible.
3.9 Arising from chair	Start from sitting in a chair, cross arms across the chest and then stand up.
3.10 Gait	Walk away from and towards the examiner. Stride amplitude, stride speed, height of foot lift, heel strike during walking, turning, and arm swing are assessed.
3.11 Freezing of gait	While gait is assessed, also assess any gait freezing episodes (start hesitation or stuttering movements), especially when turning and reaching the end of the task.

Continued on next page

Table 2.1: (continued)

Item	Description
3.12 Postural stability	While the patient is standing, the examiner stands behind the patient and pulls on the shoulders briskly and forcefully enough that the patient must take a step backwards. Number of steps to recover balance is assessed.
3.13 Posture	While standing, flexion, stooped posture, and side-to-side leaning are assessed.
3.14 Global spontaneity of movement (body bradykinesia)	Global impression of slowness and presence of spontaneous movements.
3.15 Postural tremor of hands	While arms are stretched out in front of the body, tremor presence and amplitude are assessed.
3.16 Kinetic tremor of hands	Perform at least three finger-to-nose maneuvers, reaching as far as possible to touch the examiner's finger. Tremor presence and amplitude are assessed.
3.17 Rest tremor amplitude	While sitting in a chair with hands on the arms of the chair, tremor presence and amplitude are assessed.
3.18 Constancy of rest tremor	Percentage of tremor presence over the entire examination period

2.3 Machine learning and wearable sensors for PD

The current gold standard in PD severity assessment and monitoring is the Movement Disorder Society Unified Parkinson's Disease Rating Scale (MDS-UPDRS, or UPDRS) [42]. It is periodically administered by clinicians and consists of four parts. Part I is a questionnaire that concerns non-motor aspects of daily living, such as cognitive impairment, depression, anxiety, and insomnia. Part II is a questionnaire that concerns motor aspects of daily living, *i.e.* ability to complete activities of daily living (ADL), such as speaking, eating, dressing, bathing, and walking. Part III is a motor examination that must be administered by a clinician (see Table 2.1 for a list of the questions). Part IV concerns motor complications from medication, such as side effects (dyskinesia) and fluctuations in symptom management. Each question is rated on a scale of 0 to 4, and ratings are summed to obtain a total score, either for the entire UPDRS or a particular part.

Despite being the current gold standard, the UPDRS has several weaknesses: Subjective ratings make it difficult to compare ratings from different clinicians. Because the motor portion must be administered by a clinician, the UPDRS must be done in the clinic. But because stress, time of day, and temperature can all affect a patient's symptoms, these assessments may not accurately reflect a patient's experience at home. Furthermore, it is unrealistic to use the UPDRS to monitor patients over short periods of time – for example, to understand how their symptoms progress throughout the day or respond to medication.

Wearable sensors and machine learning have great potential to improve the current state of PD assessment and monitoring. There has therefore been a preponderance of interest in automated and objective analysis of PD, evidenced by several review papers on the topic: [62, 74, 91, 113]. Many researchers have attempted to either diagnose PD or differentiate between healthy controls and PD patients through a variety of means, such as gait analysis [7, 8, 16, 38, 84, 126, 131], timed up and go test [1], typing on a cellphone keyboard [9], finger tapping [58, 77], posture [92], voice [11, 81, 120], or force control [20]. These papers are summarized chronologically in Table 2.2. There are some interesting trends to note. We can see that, in general, many research groups collect their own datasets, typically with 10-30 PD participants. Also, classification methods are often either simple thresholds on computed features, or well-known algorithms, such as SVM, multi-layer perceptrons (MLP) or random forests. In general, researchers are able to achieve classification accuracies of 90-95%. Note that, in many of the papers, the authors explored a large variety of features and/or algorithms. For brevity, only results from the best performing methods are shown in Table 2.2. Finally, we can see that as smartphones have become more ubiquitous in society, researchers have become increasingly interested in using them as the main sensing device.

Researchers have also been interested in automated and objective PD severity estimates (see Table 2.3). One group built a device in which patients applied force to two force sensors and were asked to modulate their force to track a target waveform [19, 100]. They proposed that their device could objectively measure fine changes in PD motor control, and reported significant correlation between their device and UPDRS scores. Other researchers have aimed to output UPDRS scores automatically, with the goal of making the UPDRS less labor-intensive and more objective, which could allow it to be performed at home [56, 68, 77, 78, 96, 103, 127, 130]. Similar to the work on PD classification from healthy controls, we find that wearable accelerometers are popular prior to 2014, with greater use of smartphones more recently. We also see similar dataset sizes of approximate 20 PD participants. A common way to demonstrate efficacy of predicting UPDRS is through significant correlation with UPDRS scores [68, 77, 78, 130] Other researchers formulated the task as a multi-class classification problem and report accuracies [96, 103, 127]. In [56], Heldman et al. demonstrated that their Kinesia™ system obtained higher intraclass correlation than clinicians for certain aspects of the UPDRS, which suggests that their system is more objective and consistent in its ratings. Interestingly, the authors did not report whether outputs from the Kinesia™ correlated with values assigned by clinicians.

Another application of machine learning and wearable sensors to PD management is through automated detection and/or evaluation of particular symptoms. This technology could enable in-home, continuous monitoring, which would allow patients and clinicians to better understand how symptoms change throughout the day and respond to medication. Related work on this topic are listed in Table 2.4. Note that, while many of these works report

Table 2.2: Related work on automated detection of Parkinson's Disease

	# Subjects	**Sensor(s)**	**Activities**	**Methods**	**Results**
Brewer et al. (2009) [20]	30 control 30 healthy	Device for measuring force between thumb and index	Exert force following target waveform (yes/no distraction)	Linear SVM	Acc: 85
Little et al. (2009) [81]	8 control (43 samples) 23 PD (147 samples)	Voice data (microphone)	Sustained vowel phonations	Kernel SVM	Acc: 91.4
Bakar et al. (2010) [11]	Dataset from Little et al. (2009) [81]			MLP	Acc: 92.95
Spadoto et al. (2010) [120]	Dataset from Little et al. (2009) [81]			Optimum-path forest	Acc: 75.3
Tien et al. (2010) [126]	16 control 12 PD w/ gait disturbance 11 PD w/o gait disturbance	IMU (foot)	Walk	RBF SVM	Sens.: 93.3 Spec.: 88.9
Barth et al. (2011) [16]	16 control (C) 14 early PD (E) 13 intermediate PD (I)	IMU (foot)	Walk Heel-toe tapping Circle foot	LDA (also boosting with decision stump and SVM)	C v. E: Sens.: 88 Spec.: 85 C/E v. I: Sens.: 100 Spec.: 100

Continued on next page

Table 2.2: (Automated PD detection continued)

	# Subjects	Sensor(s)	Activities	Methods	Results
Fatmehsari and Bahrami (2011) [38]	10 control 9 PD	Gyro (R arm) (other sensor locations unused)	Walk Stand still	Threshold on Lempel-Ziv Complexity measure	Sens.: 88.89 Spec.: 100 Acc: 94.74 AUC: 0.967
Hoffman and McNames (2011) [58]	35 control 11 PD	IMU (index finger)	Finger tap Pronation/ supination	Threshold on normalized mean squared error between true signal and predicted signal from forward linear prediction	AUC: 0.781-0.869
Manap et al. (2011) [84]	20 control 12 PD	37 reflective markers (Vicon) Force plate	Walk	MLP	Acc: 95.63
Palmerini et al. (2011) [92]	20 control 20 PD	3D acc (lower back)	Standing (eyes open/close) (yes/no distraction) (hard/soft floor)	SVM	Acc: 95
Weiss et al. (2011) [131]	17 control 22 PD	3D acc (lower back)	Walk	Frequency domain analysis	PD patients have lower and wider peaks than controls (statistically significant)

Continued on next page

Table 2.2: (Automated PD detection continued)

	# Subjects	**Sensor(s)**	**Activities**	**Methods**	**Results**
Adame et al. (2012) [1]	10 control (C) 10 early PD (E) 10 late PD (L)	IMU (lower back)	Timed up and go (TUG) test	DTW to segment TUG into sit-to-stand, turning, turn-to-sit	L takes longer for sit-to-stand and turning, than C and E (statistically significant)
Arora et al. (2014) [8]	10 control 10 PD	Smartphone (acc)	Walk Stand still (at home)	Random forest	Sens.: 98.5 Spec.: 97.6
Arora et al. (2015) [7]	10 control 10 PD	Smartphone (app)	Voice Stand still Walk Finger tapping Reaction time (at home)	Random forest	Sens.: 96.2 Spec.: 96.9
Lee et al. (2016) [77]	87 control 57 PD	Smartphone (app)	Two-target tapping test	Threshold on feature	AUC: 0.92
Arroyo-Gallego et al. (2017) [9]	23 control 21 PD	Smartphone (app)	Typing (5 mins/person)	Threshold on feature Linear SVM	Sens.: 0.81 Spec.: 0.81 AUC: 0.91 Sens.: 0.73 Spec.: 0.94 AUC: 0.88

Table 2.3: Related work on automated assessment on Parkinson's Disease severity

	Goal	**# subs (PD)**	**Sensor(s)/Activities**	**Methods/Results**
Van Someren et al. (2006) [130]	Automated detection/ evaluation of tremor	9	Actiwatch (acc) Regular clinic visit, UPDRS tremor severity scored for each minute	Fuzzy logic approach 0.93 corr. to UPDRS score averaged across the entire observation period Detection performance not reported
Brewer et al. (2009) [19]	Objective, sensitive measure of PD severity	26	Use index and thumb to exert force on two sensors, and modulate force to track a target waveform.	Ridge regression between output and UPDRS score. Mean absolute error = 3.58 Significant corr. with UPDRS score, $R = 0.78$
Patel et al. (2009) [96]	Automated UPDRS evaluation of tremor, bradykinesia, and dyskinesia	12	8 acc (upper/lower arms/legs) UPDRS tasks: Finger-to-nose Finger tapping Open/close hand Pronation/supination Heel tapping Quiet sitting	Multi-class (OVA) kernel SVM, trained within each subject. Classification accuracies: 96.6 for tremor, 97.8 for bradykinesia, 96.8 for dyskinesia
Pradhan et al. (2009) [100]	Develop clinical progression marker	30	Same as Brewer et al. (2009) [19]	Significant corr. with UPDRS score, $R = 0.58$

Continued on next page

Table 2.3: (Automated PD severity assessment continued)

	Goal	# subs (PD)	Sensor(s)/Activities	Methods/Results
Tsipouras et al. (2011) [127]	Automated classification of dyskinesia severity	24	6 acc (wrists, ankles, torso, waist), 2 gyros (torso, waist) Sit on chair, dyskinesia severity rated on 0-3 scale	Artificial neural network (only 1-2 sensors at a time) Average classification results across four severity values: Acc: 83.3-85.3 Sens: 61.76-73.03 Spec.: 71.86-78.18
Heldman et al. (2014) [56]	Automated UPDRS assessment	18	Kinesia, portable kinematic system (like an IMU) UPDRS tasks: Finger tapping Rest tremor Postural tremor	Kinesia has higher intraclass correlation (ICC) than clinicians for speed, amplitude, and rhythm of finger tapping. Not significantly different ICC for rest and postural tremor. Correlation of Kinesia to clinicians not reported.
Printy et al. (2014) [103]	Classification between more/less severe UPDRS (total score and bradykinesia subscore)	18	Smartphone (app) UPDRS tasks: Open/close hands Finger tapping Two-target tapping Pronation/supination	RBF SVM More/less severe UPDRS acc.: 94.5 More/less severe bradykinesia acc.: 65

Continued on next page

Table 2.3: (Automated PD severity assessment continued)

	Goal	# subs (PD)	Sensor(s)/Activities	Methods/Results
Arora et al. (2015) [7]	Automated UPDRS assessment (total score)	10	Smartphone (app) Tasks: Voice Stand still Walk Finger tapping Reaction time (at home)	Random forest Mean absolute error 1.26 UPDRS points (range 11 to 34)
Kassavetis et al. (2016) [68]	Automated UPDRS evaluation (specific tasks)	14	Smartphone (app) UPDRS tasks: Rest tremor Postural tremor kinetic tremor Pronation/supination Leg agility Finger tapping	Significant corr. for all tasks (excluding kinetic tremor), $\|R\| = 0.5$ to 0.75
Lee et al. (2016) [77]	Automated UPDRS evaluation (total score and total bradykinesia subscore)	57	Smartphone (app) Two-target tapping test	Significant corr. total score, $R = -0.5$, and bradykinesia subscore, $R = -0.32$.
Lee et al. (2016) [78]	Automated UPDRS evaluation (total score only)	103	Smartphone (app) Two-target tapping test (at home)	Significant corr. with UPDRS score, $R = 0.281$ to 0.608

Table 2.4: Related work on automated detection of Parkinson's Disease symptoms

	# Subjects	Sensor(s)/Activities	Methods/Results
Salarian et al. (2007) [111] Tremor Bradykinesia	10 control 10 PD (Part I) 11 PD (Part II)	3D gyros (wrists) Part I: 45 min. ADL Part II: 5 hr. unscripted	Tremor: Threshold on dominant frequency and amplitude Sens. = 99.5 (PD only) Spec. = 94.2 (controls only) Significant corr. with UPDRS: $R = 0.86$ (Part I), $R = 0.85$ (Part II). Bradykinesia: Threshold on gyro features Significant corr. with UPDRS: $R = -0.83$ (Part I), $R = -0.81$ (Part II).
Bachlin et al. (2010) [10] Freezing of Gait (FoG)	10 PD	3D acc (upper/lower leg and belt) Walking	Threshold on feature computed from frequency domain. Sens.: 73.1; Spec. 81.6
Cole et al. (2010) [26] Tremor Dyskinesia	4 control 8 PD	3D acc and surface electromyography (upper/lower R arm, upper R leg, lower L leg) 4 hrs. unscripted and unconstrained activities in apartment-like environment	Dynamic neural network trained on 10 mins. data from controls and 10 mins. data from PD subjects with either tremor or dyskinesia. Tremor results on left out subjects: Sens. = 87.2-91.7; Spec. = 92.0-93.0 Dyskinesia results on left out subjects: Sens. = 93.4-95.7; Spec. = 93.6-94.9

Continued on next page

Table 2.4: (Automated detection of PD symptoms continued)

	# Subjects	**Sensor(s)/Activities**	**Methods/Results**
Zwartjes et al. (2010) [143] Tremor Bradykinesia	7 control 6 PD	IMU (wrist, thigh, and foot of most affected side, and trunk) Various UPDRS tasks and ADL	Decision tree for activity recognition, then thresholds on features computed from frequency spectrum. Tremor results: Detection acc.: 78.7-94.1 Significant corr. with UPDRS scores, $R = 0.67$ to 0.84 Bradykinesia results: Significant corr. with UPDRS scores, $\|R\| = 0.57$ to 0.72 (for subset of activities)
Handojoseno et al. (2012) [51] Freezing of Gait (FoG)	26 PD	Electroencephalogram (EEG) Timed-up-and-go (TUG)	Neural network Normal v. FoG onset classification Acc.: 75.0; Sens.: 72.0; Spec.: 77.2 Normal v. FoG classification Acc.: 73.9; Sens.: 71.2; Spec.: 77.2
Sama et al. (2012) [112] Dyskinesia	20 PD	IMU (waist) Walk, set table, go up/down stairs	Thresholds on total power in 1-4Hz spectrum and under 20Hz Sens.: 0.89; Spec.:0.78
Tsipouras et al. (2012) [128] Dyskinesia	5 control 5 PD w/o dyskinesia 6 PD w/ dyskinesia	6 acc (wrists, ankles, torso, waist), 2 gyros (torso, waist) ADL, severity rated on 0-3 scale.	Decision tree Average classification results across four severity values: Sens: 80.35; Spec.: 76.84

Continued on next page

Table 2.4: (Automated detection of PD symptoms continued)

	# Subjects	**Sensor(s)/Activities**	**Methods/Results**
Cole et al. (2014) [27] Tremor Dyskinesia	Training set: 10 PD Testing set: 4 control 8 PD	3D acc and surface electromyography (dominant wrist, shin of most symptomatic leg) Scripted/unscripted ADL in apartment-like environment Tremor and dyskinesia events annotated, rated as mild, moderate, or severe	Dynamic neural network Tremor results: Detection acc.: 93.8 Severity classification: Sens.: 95.2-97.2; Spec.: 97.1-99.3 Dyskinesia results: Detection: acc.: 91.2 Severity classification: Sens.: 91.9-95.0; Spec.: 94.6-98.6
Khan et al. (2014) [71] Tremor Dyskinesia	12 PD	3D acc (waist) ADL	Multi-class (OVA) SVM with RBF kernel Classification acc. between normal, tremor, and dyskinesia: 72
Pulliam et al. (2017) [104] Tremor Dyskinesia Bradykinesia	13 PD	3D gyro and acc (most affect wrist and ankle) ADL	Linear mapping of feature computed from frequency spectrum Tremor detection: TPR: 0.90; FPR: 0.20; AUC: 0.89 Dyskinesia detection: TPR: 0.74; FPR: 0.15; AUC: 0.86 Bradykinesia detection: TPR: 0.80; FPR: 0.34; AUC: 0.82

either correlation with UPDRS scores or multi-class classification accuracies for varying levels of severity, these works differ from those in Table 2.3 in that they do so while participants perform activities of daily living (ADL), rather than specific UPDRS motor tasks. We can see that the datasets for these studies are smaller than those in Tables 2.2 and 2.3, often limited to approximately 10 PD participants. This trend is likely a reflection of the increased difficulty in labeling the data: noting the exact start and end of every symptom occurrence is much more labor-intensive than assigning a UPDRS score for a block of time or annotating whether data came from a PD or control participant. As before, methods for automated detection are often either thresholds on features or well-known algorithms, such as neural networks or decision trees. Note that all of the machine learning methods in these papers fall under fully supervised learning. That is, the algorithms assume that each segment of time is precisely annotated as symptomatic or not.

2.3.1 In-home PD assessment

While the aim of the studies listed in Table 2.4 is ostensibly automated in-home symptom detection, all the data were collected in laboratory settings (so that video data could be used for annotation). Some researchers, see [26, 27, 111], attempted to make the data more realistic by including longer time periods of unscripted activity. Nonetheless, these data may not accurately reflect a patient's at-home disease state. As such, many researchers have begun exploring systems for automated at-home disease assessment.

Mera et al. [88] evaluated the feasibility and patient compliance of using the Kinesia™ system – a wearable motion capture device – for in-home UPDRS evaluation, where 10 PD patients were asked to perform 3-6 motor assessments per day for 3-6 consecutive days in addition to annotating their medication intake times. The authors demonstrated significant differences in scores assigned pre- and post-medication intake. However, no ground-truth UPDRS scores were used for comparison in this study.

In [7], Arora et al. used a smartphone app to try to evaluate UPDRS scores at home. Participants were asked to perform specific tasks (see Table 2.3) with the smart phone four times daily for four weeks. Performance was evaluated by computing the mean absolute error (MAE) between the predicted UPDRS total scores and those given by clinicians via remote assessment once/week. The authors demonstrated an MAE of 1.26 UPDRS points (range 11 to 34).

In 2015, the mPower Mobile Parkinson Disease Study was launched with the aim of developing a large database of longitudinal motor data from healthy and PD participants using a smartphone app [18, 110]. Participants were asked to perform five tasks – walking, voice, tapping, tremor, and memory – and the app suggested that they do so three times per day. Participants also self-reported demographic information and answered questions from the UPDRS Parts I and II. Table 2.5 lists several publications utilizing the mPower dataset.

Zhan et al. [137] developed another smartphone app – HopkinsPD – for monitoring PD through active tests (voice, balance, walking, finger tapping, reaction, rest tremor, and postural tremor), passive monitoring (IMU data, GPS location, WiFi parameters, phone usage, location, social behavior), and self-reports of health, mood, and well-being. 121 PD

Table 2.5: Related work utilizing the mPower dataset [18, 110]

	Data	**Methods**	**Results**
Perumal et al. (2018) [97]	Tremor	Random Forest	Acc.: 0.68-0.85; AUC: 0.68-0.80
Pittman et al. (2018) [99]	Walking	Five different common machine learning algorithms, best performance from Multi-layer Neural Network	Healthy classification: Prec.: 0.89; Recall: 0.80; F-1: 0.84 PD classification: Prec.: 0.87; Recall: 0.93; F-1: 0.90
Prince et al. (2018) [102]	Finger Tapping Memory	Comparison of number of taps and memory score	Significant differences between healthy and PD participants for both number of taps and memory score. Significant correlation between number of taps and UPDRS scores (Parts I and II)
Prince et al. (2018) [101]	Finger Tapping	Deep Neural Network on 13 spatio-temporal features	AUC: 65.7; F-1: 61.4; Acc.: 61.2
Wroge et al. (2018) [134]	Voice	Six different common machine learning algorithms, best performance from the gradient boosted decision tree	Acc.: 0.86; F-1: 0.79; Prec.: 0.85; Recall: 0.73
Schwab and Karlen (2019) [115]	Finger Tapping Walking Voice Memory	Hierarchical neural attention mechanism over neural networks (various structures depending on the input data)	AUC: 0.85; F-1: 0.81; Sens.: 0.54 at 95% specificity

and 105 control participants enrolled and were asked to conduct the active tests twice daily (immediately before and one hour after medication for PD participants, morning and one hour later for controls). Using a random forest to classify PD participants from controls, the authors were able to obtain sensitivity, specificity, and accuracy values of 69.3, 72.7, and 71.0 respectively.

Twenty-seven participants from the Parkinson's Progression Markers Initiative (a large, longitudinal dataset collected in clinic) were asked to use the Objective Parkinson's Disease Measurement (OPDM) system, which has a similar form-factor to a laptop, during several clinic visits and on a weekly basis at home for three months [21]. Participants were asked to complete finger tapping, hand tapping, and a pegboard test using the OPDM system weekly for three months. The authors reported significant correlations with UPDRS scores.

Lipsmeier et al. [80] used a smartphone app to monitor 44 PD patients during a 6-month phase 1b clinical drug trial when they performed specific tasks on a smartphone: voice, rest tremor, postural tremor, finger tapping, balance, and gait. Additionally, 35 age-matched controls participated in a 45-day study completing the same six tasks. All participants were asked to complete the tasks once daily. The phones also collected passive data of daily activity via the smartphone sensors. The authors demonstrated significant differences between PD patients and controls for all features computed from the smartphone data. Additionally, they demonstrated significant correlations with UPDRS scores for rest tremor, postural tremor, finger tapping, balance, and gait.

These studies, all within the past five years, reflect the great potential of smartphones and other portable devices to facilitate in-home monitoring by enabling motor tests that can be performed in-home and analyzed automatically. In this thesis, wearable accelerometers were used for data collection. However, the algorithms presented in this thesis could certainly be applied to data collected from smartwatches and synchronized with a smartphone app for visualization.

2.3.2 Passive monitoring of PD symptoms

The in-home motor tests described above will enable more frequent assessment of PD symptoms. However, all required PD participants to perform multiple tasks either weekly, daily, or multiple times per day, which can be burdensome for patients. Therefore, we believe the future in PD monitoring is through passive, continuous monitoring "*in-the-wild*," where the system automatically detects symptoms through a smartphone or smartwatch without requiring any specific interaction from the user. Several studies have collected this form of naturalistic data.

Weiss et al. [131] collected in-home gait data over three days from one PD patient and one healthy control, and found that the PD patient had shorter wider peaks in the frequency spectrum compared to the control, indicating that the walking pattern of the PD patient was less consistent.

Griffiths et al. [46] collected upper-extremity data over 10 days on 34 subjects with PD and 10 age-matched controls. Using a linear combination of mean spectra power in the 0.2-4 Hz band and peak acceleration, they computed a bradykinesia score (BKS) and dyskinesia score (DKS) for each 2-minute window. On in-home data, they found that BKS scores decreased and DKS scores increased after reported medication intake. These results suggested that their algorithm was able to detect bradykiensia and dyskinesia, because medication leads to a reduction of symptoms but an increase in side effects (dyskinesia).

In an in-home study for essential tremor detection [105], 20 participants with essential tremor wore the Kinesia™ system, developed by Great Lakes NeuroTechnologies Inc., at home for ten hours per day for two days. Each hour, subjects were asked to complete three tasks to evaluate rest, postural, and kinetic tremor. These hourly assessments were processed into 0-4 scores using an algorithm that had previously been demonstrated to be correlated with clinician UPDRS scores [41]. Another algorithm based on time- and frequency-domain features (details unspecified) computed a continuous tremor severity estimate. Graphical results for one patient indicated that the continuous tremor severity estimates were similar to the hourly assessments computed when the patient completed the tremor tasks. Summary statistics for how this continuous tremor severity estimate correlated with the hourly assessments across all patients were not reported.

The large-scale Parkinson**@**home study asked nearly 1000 participants to interact with the Fox Wearable Companion app on a smartphone and smartwatch while also wearing a fall detector for six to thirteen weeks [116, 117]. Statistical analyses of this dataset indicated that higher severity of motor symptoms was associated less time spent walking [118]. It was also found that the incidence of falls was higher among PD patients than controls [119].

Zhan et al. [137] and Lipsmeier et al. [80] (described above) included passive data collection in their experimental design. However, these data were only used to differentiate PD patients from controls.

None of the data collections described in this section asked participants to label when they experienced symptoms. Therefore, none attempted to explicitly detect symptoms in the wild.

2.3.3 Weakly supervised learning for PD symptom detection

Without labels, it is difficult to build a machine learning algorithm to detect when symptoms occur. Typical machine learning algorithms require accurate labels indicating the exact start and end of every symptom occurrence. Therefore, a popular strategy is to collect data of participants performing activities of daily living while in laboratory settings, where video data can be used to label symptom occurrences (see all citations in Table 2.4). The assumption is that machine learning algorithms trained on such data will generalize (*i.e.*, perform similarly) when applied to data from patient homes. Some researchers have attempted to train their algorithms directly on data collected "in-the-wild." In these situations, accurate labels are prohibitively labor intensive to obtain. Therefore, researchers ask study participants to label approximately when their symptoms occur. These approximate labels are called *weak labels*. Chapter 3 of this thesis describes the protocol that was designed to collect weak labels in the wild while encouraging accurate and frequent labeling.

While weakly supervised learning methods are relatively well-explored (see [5] for a review), few have applied these methods to human activity recognition in time series data.

Typically, problems in this domain are formulated as multiple instance learning (MIL). That is, segments of time, which contain *multiple instances*, are labeled as containing or not containing an activity of interest. Ali and Shah [4] used MIL slightly differently, in that they represented actions in video data by several "kinematic modes" and use MIL to learn which kinematic mode represents each action. Ikizler-Cinbis and Sclaroff [63] used MIL to help combine object-, scene-, and action-related features computed from Youtube videos for improved activity recognition. Guan et al. [47] developed a generative graphical model based on an auto-regressive hidden Markov model for MIL and applied it to accelerometer data of human activity recognition. More related to the work in this thesis is that of Stikic et al. [121, 122], who used MIL for activity recognition from accelerometer data collected from real-world settings. From the annotated data, they generated three types of weak labels: the current activity, the set of activities that were performed in a given time interval, or the activity that was performed the most in a given time interval. They developed extensions to the MIL SVM algorithm to handle these different types of labels and showed that their MIL algorithms were generally able to maintain similar performance as the fully supervised algorithms, even as the time intervals increased in length. In Chapter 6 of this thesis, weak labels were also generated from accurate labels and a new type of label, called stratified weak labels, were developed specifically for the PD tremor detection application.

A small body of work has explored the use of weakly labeled data for in-home PD symptom detection. Fisher et al. [40] used a neural network to detect dyskinesia in weakly labeled data collected in the homes of 34 subjects with PD, but treated the data as accurately labeled before training the network. That is, the label assigned to each hour (either "on" medication, "off" medication, dyskinetic, or asleep) was assumed to apply to the entire hour. This assumption may be valid for the labels used in this study, but may be less accurate for symptoms like tremor, which occur more intermittently than sleeping or on/off medication state. Nonetheless, the authors reported relatively low sensitivity values (0.56), although specificity values were reasonable at 0.90. Das et al. [32] compared several weakly supervised learning techniques on in-home data collected from two subjects. While results were promising, the algorithms were trained and tested within the same subject because each subject experienced different symptoms. In our follow-up paper, and presented in Chapter 6 of this thesis, Zhang et al. [138] compared several weakly supervised learning algorithms, including a novel "stratified" algorithm, on a larger dataset collected in a laboratory setting. Chapter 7 of this thesis assesses how well laboratory data represents data collected in-the-wild. Additionally, the performance of algorithms trained on these two types of data is compared.

Chapter 3

Data Collection and Processing

This chapter describes the data collection protocol for both laboratory and in-home (*i.e.*, in-the-wild) data. A link to this dataset can be found at `http://www.humansensing.cs.cmu.edu/software`.

3.1 Laboratory recordings (LAB)

Data were collected from 12 subjects (eight male, four female, ages 66 to 85) who had been diagnosed with PD two to five years prior. Each subject self-reported tremor in one or both hands. The subjects wore one Axivity AX3 accelerometer (see Figure 3.1) on each wrist while they completed several actions, some of which were taken from the UPDRS, and others from daily living (see Table 3.1 for a complete list). To synchronize the accelerometers with each other and the cameras, each participant was asked to clap five times at the start and end of the session. Data were collected at 100 Hz. Note that subjects 1 and 6 did not exhibit any tremor during the data collection and were thus excluded from the dataset. To give the reader a sense of the overall symptom severity of our subjects, Table 3.3 shows UPDRS scores of each subject provided by a clinical expert.

Figure 3.1: Axivity AX3 accelerometer

Three cameras were used to record the subjects so as to minimize occlusion (see Figure 3.2 for views from the three cameras). To synchronize the three cameras, a light was flashed at the start of each session. Using the video recordings, tremor segments were labeled with ELAN [36]. All participants were labeled twice: In the first phase, five different people labeled all sessions holding frequent meetings for reaching labeling agreements. In the second phase, an additional person reviewed and relabeled all the sessions for consistency. The mean correlation between labels from the two phases is 0.81 with a standard deviation of 0.18. Although the correlation is high, the second phase helped to detect and correct some errors and disagreements. For this thesis, the labels from the second round were used because they

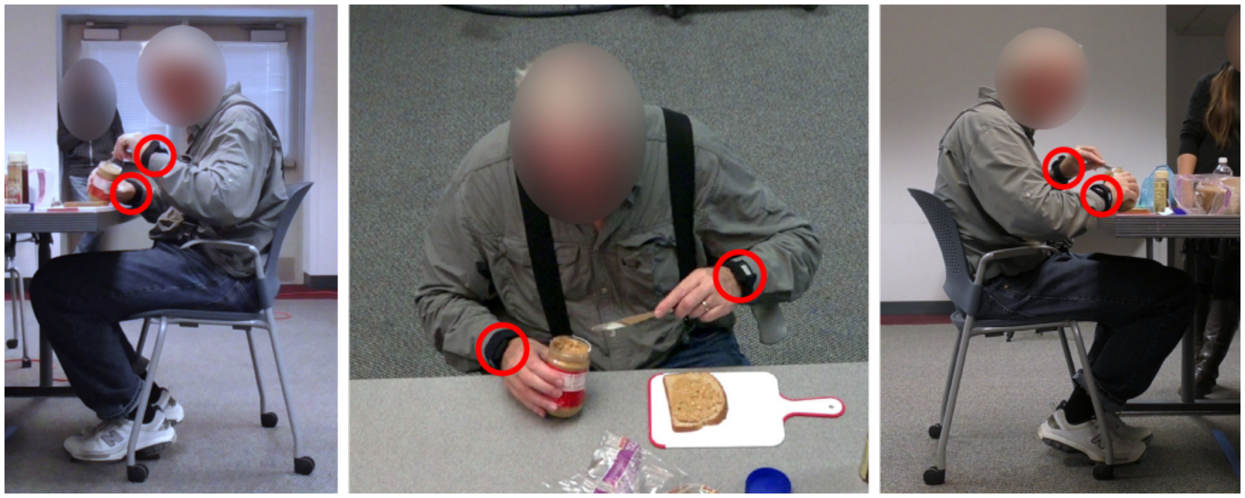


Figure 3.2: Experimental setup

Table 3.1: Set of actions performed during lab data collection

Action	Approximate time (minutes)
Sit and talk	5
Rest tremor* (UPDRS 3.17)	3
Postural tremor* (UPDRS 3.15)	6
Kinetic tremor (UPDRS 3.16)	2
Finger tapping (UPDRS 3.4)	1
Open/close hands (UPDRS 3.5)	1
Pronation/supination of the hands (UPDRS 3.6)	1
Writing	4
Typing	4
Playing chess	10
Playing cards	10
Making a sandwich	5
Eating a sandwich	10
Drinking from a cup	1
Walking	2

* denotes the inclusion of a cognitive distractor (counting backwards by 7's from 100).
Note: UPDRS item numbers correspond to those given in [42]

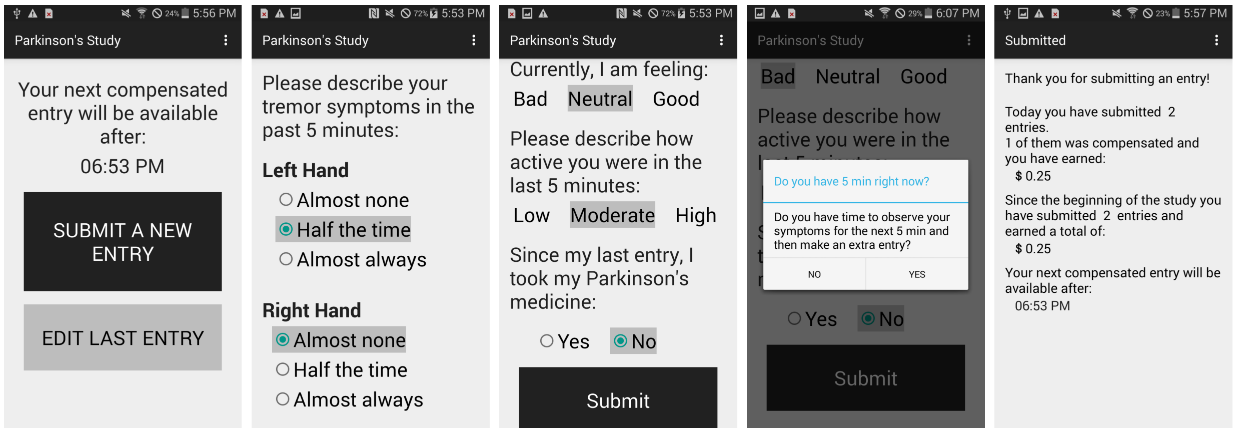
Table 3.2: Summary of data collected in lab

Subject	**# Labeled minutes per hand**	**% Tremor events**	
		Left hand	Right hand
2	74.9	80.0	40.6
3	55.9	55.9	73.7
4	55.2	57.1	37.1
5	88.1	39.0	44.1
7	97.1	26.9	19.3
8	91.3	8.6	37.9
9	96.1	21.9	7.6
10	84.5	7.8	11.7
11	51.5	69.2	26.3
12	74.7	2.0	1.2
Total	**769.2 (∼12.8 hours)**		

Table 3.3: UPDRS evaluations from lab data (provided by a medical expert)

Subject #	**MDS-UPDRS task**											
	Resting tremor (3.17)		Postural tremor (3.15)		Kinetic tremor (3.16)		Finger tapping (3.4)		Hand mov (3.5)		Pron. sup. (3.6)	
	L	R	L	R	L	R	L	R	L	R	L	R
2	2	2	2	1	1	1	1	1	1	0	3	0
3	1	2	1	2	1	2	1	3	1	2	2	2
4	3	3	2	3	1	1	3	1	3	1	4	3
5	0	0	2	2	2	2	2	1	2	1	2	2
7	0	0	1	1	1	1	1	3	2	3	3	2
8	1	1	0	1	1	1	1	2	1	2	1	2
9	0	0	0	0	0	0	3	2	-	2	-	4
10	0	2	0	0	1	0	2	3	1	1	2	1
11	3	0	3	0	1	0	3	1	1	0	1	0
12	1	1	0	0	0	0	3	2	3	2	2	2

Note: The arm of participant 9 was rigid. Therefore, the hand movement and pronation-supination tasks were skipped.


Parkinson's Study

Your next compensated entry will be available after:

06:53 PM

SUBMIT A NEW ENTRY

EDIT LAST ENTRY

Parkinson's Study

Please describe your tremor symptoms in the past 5 minutes:

Left Hand

Almost none [✓ Half the time] Almost always

Right Hand

[✓ Almost none] Half the time Almost always

Parkinson's Study

Currently, I am feeling:

Bad [✓ Neutral] Good

Please describe how active you were in the last 5 minutes:

Low [✓ Moderate] High

Since my last entry, I took my Parkinson's medicine:

Yes [✓ No]

Submit

Parkinson's Study

Bad Neutral Good

Please describe how active you were in the

Do you have 5 min right now?

Do you have time to observe your symptoms for the next 5 min and then make an extra entry?

NO YES

Yes [✓ No]

Submit

Submitted

Thank you for submitting an entry!

Today you have submitted 2 entries.
1 of them was compensated and you have earned:
$ 0.25

Since the beginning of the study you have submitted 2 entries and earned a total of:
$ 0.25

Your next compensated entry will be available after:
06:53 PM

(a) (b) (c) (d) (e)

Figure 3.3: Five screenshots from the cell phone app. (a) Home page – Allows the participant to edit the last entry or submit a new entry. Note that participants were be compensated for submitting entries at most once per hour. (b) Diary entry page – Participant describes how much tremor was experienced in the past five minutes. (c) Diary entry page continued – Participant describes general mood, activities performed and whether medication was taken since the previous entry. (d) Prompt for extra entries. (e) Submission page – Tells the participant when the next entry will be available, how many entries were submitted today and in total, and how much money was earned ($0.25 per regular entry, $1.25 per extra entry, up to $15 per day).

came from a single person and presumably had greater consistency.

Features for time-series data are typically computed over windows. Therefore, the instance-level labels described above must be converted into window-level labels. In this thesis, I chose to use three-second windows with a one-second shift. Each three-second window was labeled as a tremor window if more than 50% of the instances within were labeled as tremor. This dataset is referred to as LAB data.

3.2 In-the-wild recordings (WILD)

In previous in-home data collections performed by our lab (see Das et al. [32]), participants were asked to label the data using paper diaries. This protocol had several weaknesses: (1) for blank diary entries, it was unclear whether the participants had not experienced any symptoms or whether they simply failed to complete their diary entries, (2) labels were not well-localized in time, and (3) participants were likely completing diary entries at the end of the day, leading to probable issues with recall.

In designing the new protocol, I wanted to promote regular and frequent labeling throughout the day. Thus, I decided to pay participants per entry as opposed to paying them for simply wearing the sensors. To prevent participants from submitting many entries at one

Table 3.4: Number and type of weak labels submitted by each participant

P. #	**Left hand** Almost none	Half the time	Almost always	**Right hand** Almost none	Half the time	Almost always
2	90 (26.3%)	92 (26.9%)	160 (46.8%)	102 (29.8%)	102 (29.8%)	138 (40.4%)
4	119 (19.6%)	312 (51.5%)	175 (28.9%)	325 (53.6%)	233 (38.4%)	48 (7.9%)
5	91 (15.9%)	468 (82.0%)	12 (2.1%)	94 (16.5%)	467 (81.8%)	10 (1.8%)
10	167 (70.2%)	69 (29.0%)	2 (0.8%)	167 (70.2%)	69 (29.0%)	2 (0.8%)
11	0 (0.0%)	42 (9.1%)	419 (90.9%)	459 (99.6%)	0 (0.0%)	2 (0.4%)
12	93 (85.3%)	9 (8.3%)	7 (6.4%)	72 (66.1%)	26 (23.9%)	11 (10.1%)
Total	560 (24.1%)	992 (42.6%)	775 (33.3%)	1219 (52.4%)	897 (38.5%)	211 (9.1%)

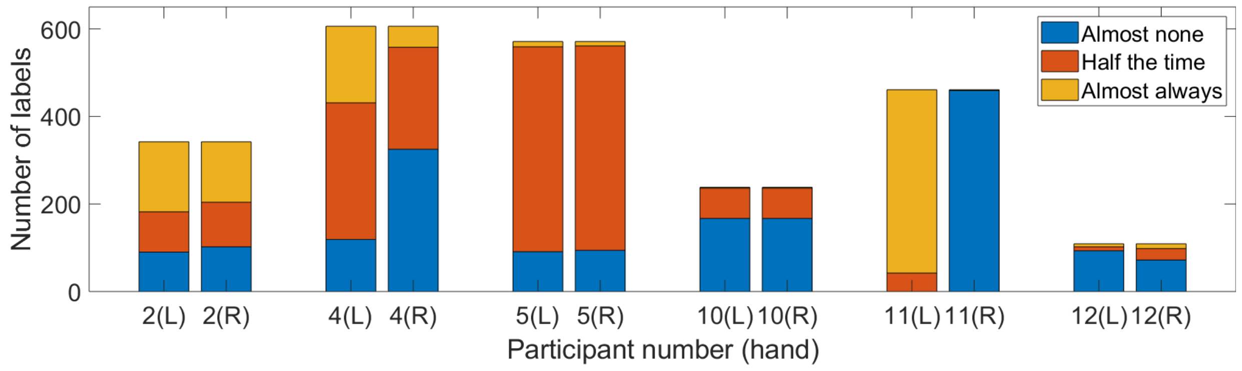


Figure 3.4: Number and type of weak labels submitted by each participant

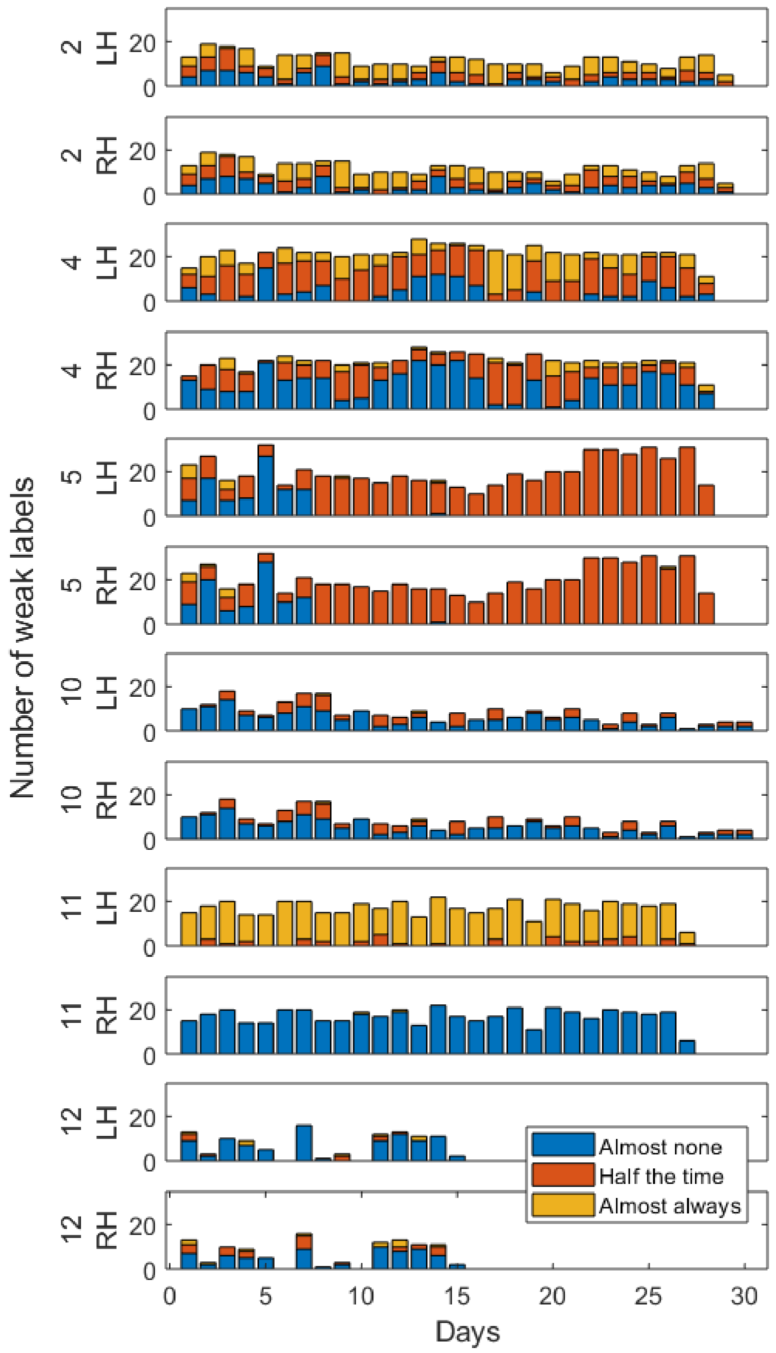


Figure 3.5: Labels submitted by each participant over time

time (all entries at the end of the day, for example), an app was designed such that participants could earn compensation ($0.25) for regular entries at most once per hour. After submitting a regular entry, participants were asked if they were available to pay attention to their symptoms for the next five minutes and submit an additional entry for $1.25. The hope was that participants would have better recall for these entries, leading them to be more accurate, because participants would specifically pay attention to their symptoms during that time. Screenshots from the app are shown in Figure 3.3. Participants were called every 1-2 days to make sure they had no problems with the sensors or the app and to keep them motivated to submit labels. Researchers also met with the participants each week to change the sensors (a maximum of two weeks of data could be stored on each sensor) and check that the data collection was progressing smoothly.

In the initial design of this study protocol, our pilot participant was asked to indicate whether or not tremor occurred since the most recent entry. However, upon reviewing the data and speaking with the participant, it was found that tremor can often go unnoticed, making it difficult to recall whether symptoms occurred in the time period between entries. Therefore, in order to promote accurate recall about their symptoms, participants were asked to record the amount of tremor they experienced only in the 5 minutes prior to submitting the entry. Given the findings presented here in Chapter 6 and published in [138], stratified, rather than binary, weak labels were used. That is, rather than asking participants whether or not they experienced tremor within the previous five minutes, participants were provided three label options (*Almost none*, *Half the time*, and *Almost always*). Additionally, while results shown Chapter 6 indicate that performance of stratified algorithms remains relatively consistent between 5- and 10-minute segments, it was expected that participant recall might more accurate for a shorter time segment.

Participants 2, 4, 5, 10, 11 and 12 agreed to participate in the in-home study. For some participants, data from their spouses were collected to serve as age-matched controls. Although these data were not used in this thesis, they have been released as part of the PD dataset. Participants wore an AX3 accelerometer on each wrist throughout the day for two to four weeks. All participants made regular entries during the data collection period: Table 3.4 and Figure 3.4 show the distribution of labels for each participant. Figure 3.5 shows the labels provided by the participants over time. In general, participants maintained their labeling frequency throughout the study, although participant 10 submitted fewer labels per day after week one. Note that, after two weeks, participant 12 decided to withdraw from the study due to being too busy or stressed to continue submitting entries. All others participants had no issues with the cell phone app or the study in general and remained in the study for all four weeks. Most participants elected to remove the sensors while sleeping, and these segments of time were manually removed from the dataset by looking at the accelerometer data. These weakly labeled data are subsequently referred to as *WILD*.

Comparing the distribution of labels in the WILD dataset (Figure 3.4) with the percentage of tremor events annotated in the laboratory data (Table 3.2), we can see that the weak labels given by the participants are generally consistent with the labels assigned by researchers on the laboratory data. For example, participants 4 and 11 indicated that tremor occurred more frequently in the left hand compared to the right, which is consistent with the higher percentage of tremor in the left hand in the LAB dataset. Participant 5 reported low to moderate amounts of tremor at home and experienced tremor approximately 40% of the time during the laboratory session. Participants 10 and 12 primarily reported "*Almost none*" for their tremor symptoms and also demonstrated a low percentage of tremor in the LAB dataset. One inconsistency between the two datasets is that Participant 2 indicated that tremor occurred equally between the hands while researchers noted 80.0% tremor in the left hand versus 40.6% tremor in the right. It is possible that this inconsistency is due to researchers including low intensity tremor that the participant does not notice at home. It is also possible that the participant perceived tremor to occur equally between the hands, even if the right hand actually tremors less. Another artifact of the dataset is the highly homogeneous labels submitted by participant 5 after week one. It is unclear whether these labels reflect the participant's true symptom experience or whether the homogeneity was due to a misunderstanding of label interpretation. Nonetheless, the similarities of label proportions between the LAB and WILD indicate that participants and researchers generally agreed on what constitutes tremor.

3.3 General training and testing procedures

All experiments in this thesis, unless specifically stated as otherwise, are conducted under leave-one-subject-out (LOSO) cross validation. That is, the algorithms are trained on the data of all subjects excluding one, and then tested on the left out subject. Note that, because people use their left and right hands in very different ways, and because symptoms of PD often manifest differently on the two sides of the body, data from separate hands were considered to be from separate subjects. This design consideration affects the STM and KMM algorithms in Chapter 4, which use unlabeled data from the test subject. It also affects TPT (Chapter 4) and the person-specific models presented in Chapter 7, which build models for individual subjects. Note however, that when leaving data out from the training set, data from *both* hands are excluded.

3.4 Features

Previous work on automated tremor detection [27, 71, 96, 98, 111] have generally used very similar features. In this thesis, I use the same features as those described by Patel et al. [96]. The accelerometer signals were first high-pass filtered with a 1 Hz cutoff using an 8th-order elliptic filter. Twenty-one features were computed over three-second windows of the signal with one-second window shift. On each window, the range, root mean square, Shannon entropy, dominant frequency and ratio of energy in the dominant frequency over total energy were computed over each axis. The peak normalized cross-correlation and the time lag associated with the peak were computed over each pair of axes. Figures 3.6 and 3.7 show example accelerometer data (after high-pass filtering) for non-tremor and tremor motion, respectively. Notably, Figure 3.6 (bottom) shows how, for voluntary motion, the energy in the frequency domain is spread throughout the 0-5 Hz band. In contrast, for tremor motion, most of the energy is concentrated at a particular, higher frequency (approximately 5Hz for this participant). Table 3.5 gives the mathematical definitions for all features and feature values for the non-tremor and tremor time segments shown in Figures 3.6 and 3.7. As expected, tremor shows larger values in both the dominant frequency and the ratio of energy in the dominant frequency over total energy. Tremor also shows higher peak cross-correlation values.

3.5 Performance metrics

This section defines several common performance metrics that are used throughout this thesis.

$$\begin{aligned}
\text{Accuracy} =& \frac{\text{\# correctly classified}}{\text{\# events}}, && (3.1)\\
\text{F1-score} =& 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}, && (3.2)\\
\text{Precision} =& \frac{\text{\# true positives}}{\text{\# classified as positive}}, && (3.3)\\
\begin{array}{r}\text{Recall/Sensitivity/}\\ \text{True Positive Rate (TPR)}\end{array} =& \frac{\text{\# true positives}}{\text{\# ground truth positive events}}, && (3.4)\\
\text{Specificity} =& \frac{\text{\# true negative}}{\text{\# ground truth negative events}}, && (3.5)\\
\begin{array}{r}\text{Area Under the ROC}\\ \text{Curve (AUC)}\end{array} =& \int_{x=0}^{1} TPR(FPR^{-1}(x))\, dx && (3.6)\\
& \text{where } FPR \text{ is the false positive rate,} &&\\
& \textit{i.e.}, \text{ \# false positives / \# classified as negative.} &&
\end{aligned}$$

3.6 Machine learning challenges on this dataset

In this dataset, it is difficult to maintain high performance on subjects who were not in the training population due to high variability between people. Tremor, for example, manifests differently between people – varying in both amplitude and frequency, and people respond differently to medication. Environmental factors can also affect a person's tremor: some of our subjects reported that they experienced worse tremor when cold or under stress. Finally, people have different lifestyles and hobbies, which leads to variability in the set of activities performed by each subject for the WILD dataset.

It is also difficult to generalize from the LAB dataset to the WILD dataset due to differences between the two. The LAB dataset protocol was designed to include many different activities of daily living, but did not include all possible activities. Our participants engaged

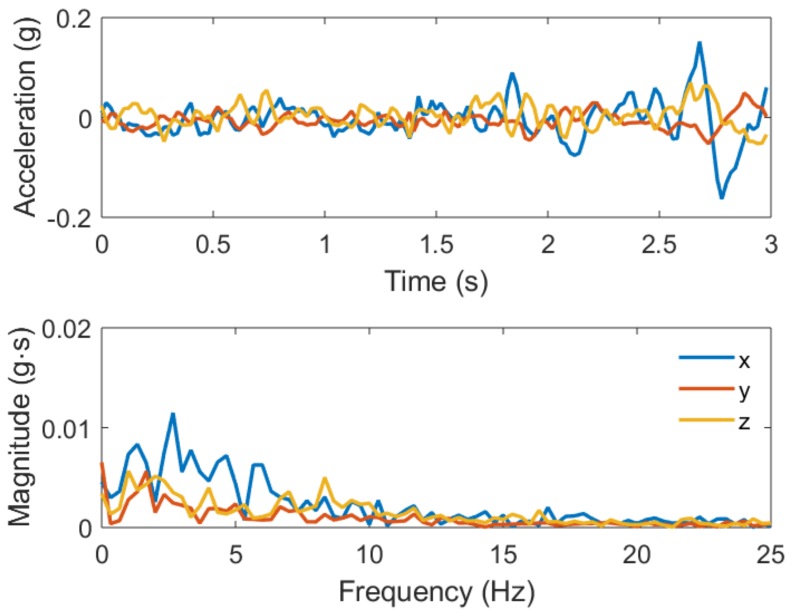


Figure 3.6: Example accelerometer data from a 3-second window of voluntary (non-tremor) motion. Time domain (top) and frequency domain (bottom).

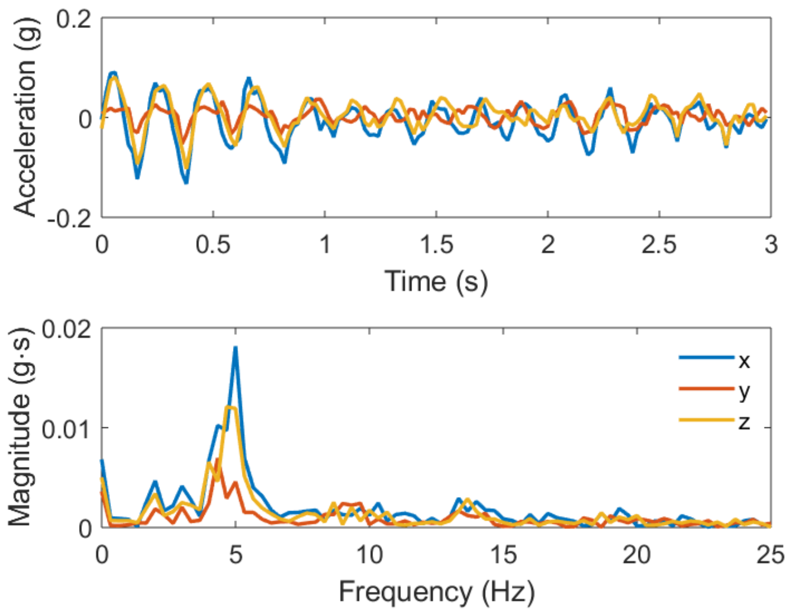


Figure 3.7: Example accelerometer data from a 3-second window of tremor motion. Time domain (top) and frequency domain (bottom).

Table 3.5: Feature definitions and example values for non-tremor and tremor

Feature	**Definition**	**Axis/ Axes**	**Example value for non-tremor (Figure 3.6)**	**Example value for tremor (Figure 3.7)**
		x	0.32	0.22
Range	$\max f(t) - \min f(t)$	y	0.10	0.09
		z	0.12	0.18
		x	0.04	0.04
Root mean square (RMS)	$\sqrt{\frac{1}{n}\sum_{t=1}^{n}(f(t))^2}$	y	0.02	0.02
		z	0.02	0.03
	$-\sum_i p_i \log_b p_i$	x	1.30	1.38
Shannon entropy	p_i indicates the probability	y	0.32	0.34
	of a particular value of the signal	z	0.60	0.89
		x	2.67	5.00
Dominant frequency	$\arg\max_{\xi} \hat{f}(\xi)$	y	0	4.33
		z	1	4.67
		x	0.07	0.13
Ratio of energy	$\frac{\max_\xi \hat{f}(\xi)}{\sum_\xi \hat{f}(\xi)}$	y	0.09	0.10
		z	0.05	0.11
		x, y	0.03	0.05
Peak cross-correlation(f, g)	$\max_{\tau} \sum_{t=-\infty}^{\infty} f(t-\tau)g(t)$	y, z	0.01	0.04
		x, z	0.07	0.16
		x, y	0.04	0
Lag associated with peak	$\arg\max_{\tau} \sum_{t=-\infty}^{\infty} f(t-\tau)g(t)$	y, z	-0.13	0.54
		x, z	-0.02	0

Note: $f(t)$ is the (discrete) time-domain signal and $\hat{f}(\xi)$ is the signal in the frequency domain.

in a variety of hobbies, such as gardening, golf, or boxing, and these activities were not included in the LAB protocol. Additionally, the proportion of time spent on each activity in the LAB dataset differs from that in the WILD dataset. For example, approximately one fourth of the LAB protocol was devoted to making and eating a sandwich, but people do not typically spend one fourth of their waking hours eating. While modifying the LAB protocol could mitigate some of these differences, recording a person's motions in a laboratory setting makes LAB data fundamentally different from WILD data.

3.7 Lessons learned

The WILD dataset presented here represents a novel protocol designed to increase label frequency and accuracy compared to the previous standard of paper diaries. However, the protocol can still be improved on several dimensions.

One weakness is the small dataset: only 6 people participated in the in-home portion, and, on average, each participant only labeled approximately 36 hours of data per hand out of the 4-week study ($\sim$ 443 samples/participant $\times$ 5 minutes/sample). In general, in-home datasets are difficult to collect because they are more labor intensive for both the participants and the researchers. Participants must submit frequent labels of their symptoms throughout the day for many weeks, and researchers must periodically contact the participants to make sure there are no problems and to exchange sensors. A larger number of participants would make performances differences between algorithms clearer. While most datasets in this domain are relatively small (see Table 2.4), Fisher et al. [40] – the most similar dataset to the one presented here – were able to collect from 34 participants. The Parkinson**@**home study did not ask participants to label their symptoms, but the nearly 1000 participants were asked to interact with a smartphone app and wear a smartwatch. Training on cohorts of this size should enable higher performance from machine learning algorithms, and more reliable conclusions could be drawn across subjects.

The data collection and algorithm training/testing protocols were designed to best evaluate generalization across time. Participants submitted labels for four weeks, which is longer than many other WILD datasets. For example, Fisher et al. [40], Das et al. [32], Pulliam et al. [105], and Griffiths et al. [46] all collected data for fewer than ten days. Temporal consistency was maintained while partitioning this dataset so that algorithms were not shown samples across the whole dataset during training. Nonetheless, it is unclear how the learned models would perform six or twelve months later. Over these longer time-spans, it is possible that a person's tremor or activities could change and therefore reduce a model's tremor detection performance. Meanwhile, a product's performance would need to be stable across months or years, which means that more longitudinal data should be collected for training and evaluating algorithms. To do so, participants could be asked to label shorter periods of time spread over a year, *i.e.*, one week every three months.

Anecdotal observations from this study show that several design improvements could be made with respect to labeling. For example, creating the additional label of "no tremor" would facilitate machine learning because it would be known that all samples within those time segments would be non-tremor. In contrast, for the “Almost none” label, only 66% of the samples are *assumed* to be non-tremor. Participants could be better informed about how to interpret “Almost none” versus “Half the time” versus “Almost always.” In this thesis, these labels are assumed to correspond to [0-33%], [33-66%], and [66-100%] tremor respectively, but it is possible that the participants interpreted the labels differently. Finally, incorporating active learning into the study design, so that the smartphone app would prompt the participants for labels of particular interest, could reduce the number of labels necessary for training.

A final area of improvement for the study protocol is in the sensors used for data collection. At the time of study design, existing commercially available wearable devices were typically limited to one sensing modality. More recently, smartwatches have been released that not only include a full inertial measuring system within, but also can measure heart rate. Substituting the AX3 sensors used here with this newer smartwatches could enable interesting insights between PD and heart rate without requiring participants to wear more sensors.

Chapter 4

Personalization Via Domain Transfer

One of the challenges in automated Parkinson's Disease (PD) symptom detection is human variability. Gender, age, and body type all affect how a person moves. People also differ in their hobbies and level of activity. PD patients in particular are all affected differently by the progression of the disease. These differences lead to high variability across people in motion data from accelerometers, while data from individuals may be clustered in distinct styles. Standard machine learning algorithms often struggle under this scenario because they assume that the test data are sampled from the same distribution as the training data, which implies that minimizing loss on the training data approximates minimizing loss on the test data. However, with human data, an individual's style is not typically sampled uniformly from the population's distribution of all possible styles.

Classifiers trained uniformly on many different subjects may become confused by these individual differences, leading to poor generalization on a test subject who was not in the training population, even if the test subject's data are highly separable and even if there exist other subjects who are similar to the test subject. Indeed, it has been shown that classifiers trained specifically on the test subject's data tend to perform much better than those trained on a population of other subjects [132]. However, such person-specific classifiers need labeled data from the test subject, and these data are often difficult to obtain. For example, users of an activity recognition product would generally expect it to work immediately out of the box and might find it cumbersome if a product required them to label their own data before a person-specific activity recognition model could be built.

While labeled data from end users can be difficult to obtain, *unlabeled* data is almost trivial to obtain. It is possible, therefore, to leverage knowledge of the distribution of a test subject or end user's data and personalize a generic classifier to perform better on that data. This process is called domain transfer. In particular, modeling the differing distributions of the training and test data falls under the covariate shift problem. Chapter 2.2 describes related work in this domain.

Several algorithms for solving the covariate shift problem have been proposed. However, few have been applied to accelerometer data, and none were applied to the problem of PD tremor detection. In this chapter, I compare a generic Support Vector Machine (SVM) to three algorithms designed to personalize an SVM given unlabeled data from a test subject:

Kernel Mean Matching (KMM) [45], Selective Transfer Machine (STM) [24], and Transductive Parameter Transfer (TPT) [114]. These algorithms were chosen for several reasons:

- They are relatively easy to understand;
- They are based on the SVM algorithm, which has been shown to have high performance in many domains, but the underlying methods of personalization can be applied to other classification algorithms; and
- They are interpretable, which is particularly desirable when building machine learning algorithms for medical applications.

4.1 Algorithms

In this section, I review the formulations of a generic SVM and the three domain transfer algorithms I compared (KMM, STM, and TPT).

4.1.1 Support Vector Machine

Let $\mathcal{D}^{\text{tr}} = \{\mathbf{x}_i, y_i\}_{i=1}^{n_{\text{tr}}}$ be a set of n_{tr} training points where $y_i \in \{+1, -1\}$ indicates the class of the sample $\mathbf{x}_i$. This algorithm tries to find a hyperplane defined by $\mathbf{w}$ and b that separates the positive and negative classes while maximizing the distance between the hyperplane and the nearest training points (support vectors). It outputs a classification function $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$, such that when $f(\mathbf{x}) > 0$, the test sample is classified as $+1$ and vice versa. $\mathbf{w}$ and b are chosen by minimizing the following objective function:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{2}\sum_{i=1}^{n_{\text{tr}}} L(y_i, f(\mathbf{x}_i)), \tag{4.1}$$

where $L(y_i, f(\mathbf{x}_i))$ represents the loss function. Many loss functions exist, but here the squared hinge-loss is used: $L(y_i, f(\mathbf{x}_i)) = \max\{1 - y \cdot f(\mathbf{x}_i), 0\}^2$. C represents the trade-off between minimizing the loss and maximizing the margin. The LIBLINEAR library written by Fan et al. [37] was used to train the SVM.

4.1.2 Kernel Mean Matching

Kernel Mean Matching (KMM), developed by Gretton et al. [45], is a personalization approach that uses importance reweighting to optimize the decision function for the test data without knowing the labels, thereby creating a classifier that is personalized for the test data. The main idea behind KMM is to weight the training data such that the weighted training distribution approximates the distribution of the test data. The assumption is that classifiers trained on a distribution that matches the test data will better generalize to the test data. Note that, in contrast to non-personalized algorithms, KMM requires test data to be available during training and must be retrained for each test subject.

To formulate KMM, I introduce some additional notation. Let $\mathbf{X}^{\text{te}} = \{\mathbf{x}_i\}_{i=1}^{n_{\text{te}}}$ be the set of n_{te} *unlabeled* test points from a single test subject. Furthermore, let $\mathbf{X}^{\text{tr}} = \{\mathbf{x}_i\}_{i=1}^{n_{\text{tr}}}$ be the set of n_{tr} training samples in $\mathcal{D}^{\text{tr}}$ after discarding the labels. Let $\mathbf{s} \in \mathbb{R}^{n_{\text{tr}}}$ be a vector that contains the positive weights s_i for each training sample $\mathbf{x}_i$. KMM aims to find sample weights s_i that reduce the mismatch between the weighted training distribution $\mathbf{X}^{\text{tr}}$ and the test distribution $\mathbf{X}^{\text{te}}$ by minimizing the distance between the empirical mean of the weighted training set and the test set in the Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$ [45]:

$$\begin{aligned} \min_{\mathbf{s}} \mathbf{\Omega}_{\mathbf{s}}(\mathbf{X}^{\text{tr}}, \mathbf{X}^{\text{te}}) &= \left\| \frac{1}{n_{\text{tr}}} \sum_{i=1}^{n_{\text{tr}}} s_i \varphi(\mathbf{x}_i^{\text{tr}}) - \frac{1}{n_{\text{te}}} \sum_{j=1}^{n_{\text{te}}} \varphi(\mathbf{x}_j^{\text{te}}) \right\|_{\mathcal{H}}^2 , \\ &= \frac{1}{{n_{\text{tr}}}^2} \mathbf{s}^\top \boldsymbol{K} \mathbf{s} - \frac{2}{{n_{\text{tr}}}^2} \boldsymbol{\kappa}^\top \mathbf{s} + \text{const.} \\ \text{subject to} \quad & \\ & s_i \in [0, B], \\ & \left| \frac{1}{n_{\text{tr}}} \sum_{i=1}^{n_{\text{tr}}} s_i - 1 \right| \leq \varepsilon. \end{aligned} \tag{4.2}$$

$\varphi(\cdot)$ represents the implicit mapping from the feature space into the Hilbert space, $\mathbf{s} = [s_1, \ldots, s_{n_{\text{tr}}}]$ is a vector containing the sample weights, $\boldsymbol{K}$ is the kernel matrix with $K_{ij} = k\left(\mathbf{x}_i^{\text{tr}}, \mathbf{x}_j^{\text{tr}}\right)$, and $\boldsymbol{\kappa}$ is a vector with $\kappa_i = \frac{n_{\text{tr}}}{n_{\text{te}}} \sum_{j=1}^{n_{\text{te}}} k\left(\mathbf{x}_i^{\text{tr}}, \mathbf{x}_j^{\text{te}}\right)$. In this thesis, the Radial Basis Function (RBF) kernel is used:

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2} \right). \tag{4.3}$$

The first constraint of Eq. 4.2 limits the influence (s_i) of individual points and the second enforces the *average* weighting on points to be within ε of 1. This objective function is a quadratic programming problem, which is solved using the MOSEK ApS optimization software [90].

Given the learned weights $\mathbf{s}_i$, one can train a classifier on the newly weighted training set. Many different classifiers can be used, but here I present the weighted SVM:

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 + \frac{C}{2} \sum_{i=1}^{n_{\text{tr}}} s_i L(y_i, f(\mathbf{x}_i)). \tag{4.4}$$

As before, the loss function $L(y_i, f(\mathbf{x}_i))$ can be any loss function, such as the squared hinge loss $L(y_i, f(\mathbf{x}_i)) = \max\{1 - y \cdot f(\mathbf{x}_i), 0\}^2$. Note how this objective function is the same as a standard SVM (Eq. 4.1) except that the loss of each sample is weighted by s_i. As before, the LIBLINEAR library [37] was used to solve this weighted SVM problem.

4.1.3 Selective Transfer Machine

While KMM does not take sample labels into consideration during training, the Selective Transfer Machine (STM), built by Chu et al. [24], attempts to leverage sample labels by finding a set of weights s_i that not only minimizes the mismatch between the training and test set distributions, but also is consistent with and can support a classification model. STM is formulated as minimizing the following objective:

$$g(f, \mathbf{s}) = \min_{f,\mathbf{s}} R_f(\mathcal{D}^{\text{tr}}, \mathbf{s}) + \lambda \mathbf{\Omega}_{\mathbf{s}}(\mathbf{X}^{\text{tr}}, \mathbf{X}^{\text{te}}). \tag{4.5}$$

$\mathbf{\Omega}_{\mathbf{s}}(\mathbf{X}^{\text{tr}}, \mathbf{X}^{\text{te}})$ is given in 4.2 and λ controls the trade-off between minimizing the distribution mismatch versus finding a classifier and set of weights that are consistent with each other. The first term, $R_f(\mathcal{D}^{\text{tr}}, \mathbf{s})$, is the SVM empirical risk defined on the decision function f and the training set $\mathcal{D}^{\text{tr}}$, with the loss of each instance $\mathbf{x}_i$ weighted by s_i. While non-linear decision functions can be used, linear methods are compact and fast. Furthermore, they may be less likely to overfit on smaller datasets. Choosing f to be a linear function, $f(\mathbf{x}_i) = \mathbf{w}^\top \mathbf{x} + b$, leads $R_f(\mathcal{D}^{\text{tr}}, \mathbf{s})$ to simply be a weighted linear SVM, as shown in 4.4:

$$R_f(\mathcal{D}^{\text{tr}}, \mathbf{s}) = R_{\mathbf{w},b}(\mathcal{D}^{\text{tr}}, \mathbf{s}) = \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{2}\sum_{i=1}^{n_{\text{tr}}} s_i L(y_i, f(\mathbf{x}_i)). \tag{4.6}$$

As before, L is chosen to be the squared hinge-loss function:

$$L(y_i, f(\mathbf{x}_i)) = \max\{1 - y \, \cdot f(\mathbf{x}_i), 0\}^2. \tag{4.7}$$

To solve STM, the weights are first initialized to the KMM solution. When solving for $\mathbf{s}$, STM differs from KMM (Eq. 4.2) only in the addition of an extra linear term:

$$\begin{aligned}
\min_{\mathbf{s}} g(f, \mathbf{s}) &= \min_{\mathbf{s}} R_f(\mathcal{D}^{\text{tr}}, \mathbf{s}) + \lambda \mathbf{\Omega}_{\mathbf{s}}(\mathbf{X}^{\text{tr}}, \mathbf{X}^{\text{te}}) \\
&= \min_{\mathbf{s}} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{2}\sum_{i=1}^{n_{\text{tr}}} s_i L(y_i, f(\mathbf{x}_i)) + \lambda \left(\frac{1}{{n_{\text{tr}}}^2}\mathbf{s}^\top \boldsymbol{K}\mathbf{s} - \frac{2}{{n_{\text{tr}}}^2}\boldsymbol{\kappa}^\top \mathbf{s} + \text{const.} \right) \\
&= \min_{\mathbf{s}} \frac{1}{{n_{\text{tr}}}^2}\mathbf{s}^\top \boldsymbol{K}\mathbf{s} + \left(\frac{C}{2\lambda}\boldsymbol{\ell} - \frac{2}{{n_{\text{tr}}}^2}\boldsymbol{\kappa} \right)^\top \mathbf{s} + \text{const.}
\end{aligned} \tag{4.8}$$

where $\boldsymbol{\ell}$ is a vector such that $\ell_i = L(y_i, \mathbf{w}^\top \mathbf{x}_i)$. This objective function is only a slight modification of Eq. 4.2, and it is also solved using the MOSEK ApS optimization software [90]. Looking at the linear term, we can see that the solution to this function increases the value of indices s_i that correspond to training points with low SVM loss l_i (*i.e.*, those that are consistent with the current classifier) and high "similarity" with the test points κ_i, where similarity is measured using the kernel function k. Furthermore, the quadratic term, which is equal to $\sum_{ij} s_i \, k(x_i^{\text{tr}}, x_j^{\text{tr}}) \, s_j$, enforces some sparsity by reducing the weight on training points that have high similarity to each other, particularly if one of them already has high weight from the linear term.

Given the current weight vector $\mathbf{s}$, the separating hyperplane $\mathbf{w}$ is found by minimizing the weighted linear SVM (Eq. 4.6), using the LIBLINEAR library [37]. The algorithm iterates between fixing $\mathbf{w}$ and solving for $\mathbf{s}$, and fixing $\mathbf{s}$ and solving for $\mathbf{w}$ until convergence, which is defined to be when the change in $\mathbf{w}$ or $\mathbf{s}$ is sufficiently small.

In summary, STM finds f and $\mathbf{s}$ to simultaneously minimize a weighted SVM empirical risk $R_f(\mathcal{D}^{\text{tr}}, \mathbf{s})$ and a measure of distribution mismatch between the weighted training set and the test set $\boldsymbol{\Omega}_{\mathbf{s}}(\mathbf{X}^{\text{tr}}, \mathbf{X}^{\text{te}})$. Figure 5.1 in Chapter 5.1.3 gives an example of how the separating hyperplane changes over iterations of STM. Note that the first iteration represents the KMM solution. $\lambda > 0$ indicates the trade-off between the SVM empirical risk and the distribution mismatch. For more details, I refer the reader to [24].

4.1.4 Transductive Parameter Transfer

While KMM and STM use importance reweighting to solve the covariate shift problem, the Transductive Parameter Transfer (TPT) algorithm, developed by Sangineto et al. [114], attempts to learn a regression from a distribution to a classifier. Suppose there are N labeled datasets from N different subjects $\mathcal{D}_1^{\text{tr}}, \ldots, \mathcal{D}_N^{\text{tr}}$, where $\mathcal{D}_i^{\text{tr}} = \{\mathbf{x}_j, y_j\}_{j=1}^{n_{tr}^i}$. As before, let $\mathbf{X}^{\text{te}} = \{\mathbf{x}_i\}_{i=1}^{n_{\text{te}}}$ be a set of n_{te} unlabeled test points from a single test subject. Similarly, let $\mathbf{X}_i^{\text{tr}} = \{\mathbf{x}_j\}_{j=1}^{n_{tr}^i}$ be the set of points in $\mathcal{D}_i^{\text{tr}}$ after discarding the labels.

The first step of TPT is to learn a linear SVM for each subject. The parameters $\mathbf{w}_i$ and b_i are trained on a single subject's dataset $\mathcal{D}_i^{\text{tr}}$ and combined into a single vector $\boldsymbol{\theta}_i = \begin{bmatrix}\mathbf{w}_i^\top & b_i\end{bmatrix}$, which represents the separating hyperplane. It is assumed that each $\boldsymbol{\theta}_i$ is dependent on the distribution P_i^{tr} from which the samples $\mathbf{X}_i^{\text{tr}}$ were drawn. Although the distribution P_i^{tr} is unknown, it can be approximated from the set of samples $\mathbf{X}_i^{\text{tr}}$. Therefore, the goal is to learn the mapping $f(\cdot)$ from datasets to separating hyperplanes. Then, given the test subject's samples $\mathbf{X}^{\text{te}}$, we can apply the mapping and obtain a hyperplane: $f(\mathbf{X}^{\text{te}}) = \boldsymbol{\theta}_{\text{te}} = \begin{bmatrix}\mathbf{w}_{\text{te}}^\top & b_{\text{te}}\end{bmatrix}$.

Sangineto et al. [114] formulated the problem of learning the mapping $f(\cdot)$ as a regression problem, and solved it using the Multioutput Support Vector Regression (M-SVR) framework proposed by Tuia et al. [129]. In particular, they defined $f(\cdot)$ by the set of parameters $\boldsymbol{\pi} = (\mathbf{B}, \mathbf{c})$:

$$f(\mathbf{X}) = \varphi(\mathbf{X})^\top \mathbf{B} + \mathbf{c},$$

where $\mathbf{B} = [\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_{M+1}]$ is the weight matrix, $\mathbf{c} = [c_1, \ldots, c_{M+1}]$ is the bias vector, and $\varphi(\cdot)$, as with KMM, represents the implicit mapping from the feature space to the Hilbert space. M is the dimensionality of the samples, *i.e.*, the number of features in the feature vector. $\boldsymbol{\pi}$ is found by minimizing the following objective function:

$$\min_{\boldsymbol{\pi}} \frac{1}{2} \sum_{i=1}^{M+1} \|\boldsymbol{\beta}_i\|^2 + \lambda \sum_{j=1}^{N} E(\|\boldsymbol{\theta}_j - f_{\boldsymbol{\pi}}(\mathbf{X}_j^{\text{tr}})\|), \tag{4.9}$$

where $E(\cdot)$ is a loss function for ε-insensitive loss used by SVR:

$$E(u) = \begin{cases} 0 & u < \varepsilon \\ (u-\varepsilon)^2 & u \geq \varepsilon \end{cases} \tag{4.10}$$

Note the similarity of Eq. 4.9 to the SVM objective function Eq. 4.1. The main difference is in the loss function, where SVM measures the "amount" that a sample is misclassified, and SVR measures the distance between the desired vector and that output by the current mapping $f(\cdot)$.

To avoid explicitly representing the non-linear mapping $\varphi(\cdot)$, Tuia et al. [129] use the representer theorem to express the problem as a linear combination of the training points $\mathbf{X}_i^{\text{tr}}$:

$$\boldsymbol{\beta}_i = \sum_{j=1}^{N} \varphi(\mathbf{X}_j^{\text{tr}})^\top \mathbf{v}_i \tag{4.11}$$

With this substitution, it turns out that each $\mathbf{v}_i$ and c_i can be solved as follows:

$$\begin{bmatrix} \mathbf{AK}+\mathbf{I} & \mathbf{a} \\ \mathbf{a}^\top\mathbf{K} & \mathbf{1}^\top\mathbf{a} \end{bmatrix} \begin{bmatrix} \mathbf{v}_i \\ c_i \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Theta}_i \\ \mathbf{a}^\top\boldsymbol{\Theta}_i \end{bmatrix}, \tag{4.12}$$

where $\boldsymbol{\Theta} = [\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_N] \in \mathbb{R}^{M+1\times N}$ and $\boldsymbol{\Theta}_i$ denotes the ith row of $\boldsymbol{\Theta}$. $\mathbf{K}$ is the kernel matrix, where $\mathbf{K}_{ij} = k(\mathbf{X}_i^{\text{tr}}, \mathbf{X}_j^{\text{tr}})$, and $k(\cdot,\cdot)$ is the kernel function, which measures the similarity between the two input distributions and which is described in detail below. $\mathbf{I}$ is the N dimensional identity matrix. $\mathbf{a} = [a_1, \ldots, a_N]$ and $\mathbf{A}$ is a diagonal matrix with elements $a_1, \ldots, a_N$. At each iteration, each a_i is computed as follows:

$$a_i = \begin{cases} 0 & u_i < \varepsilon \\ \frac{2\lambda(u_i-\varepsilon)}{u_i} & u_i \geq \varepsilon \end{cases}. \tag{4.13}$$

u_i is the norm of the error vector for the ith subject's data. That is,

$$\begin{aligned} u_i &= \left\|\boldsymbol{\theta}_i - f(\mathbf{X}_i^{\text{tr}})\right\|, \\ &= \left\|\boldsymbol{\theta}_i - \left(\varphi(\mathbf{X}_i^{\text{tr}})^\top\mathbf{B} + \mathbf{c}\right)\right\|, \\ &= \left\|\boldsymbol{\theta}_i - \left(\sum_{j=1}^{N} k(\mathbf{X}_i^{\text{tr}}, \mathbf{X}_j^{\text{tr}})\,[\mathbf{v}_1, \ldots, \mathbf{v}_{M+1}] + \mathbf{c}\right)\right\|, \end{aligned}$$

where the final equality comes from applying Eq. 4.11.

The learned values $[\mathbf{v}_1, \ldots, \mathbf{v}_{M+1}]$ and $\mathbf{c}$ are computed over iterations. Convergence conditions were set to be either (a) when the change in $[\mathbf{v}_1, \ldots, \mathbf{v}_{M+1}]$ and $\mathbf{c}$ was sufficiently small, (b) when $a_i = 0, \forall i$ (which indicates that, $\forall i$, the error $u_i < \varepsilon$ and leads the matrix on the left of Eq. 4.12 to become singular), or (c) when the number of iterations exceeded 100. Once converged, $[\mathbf{v}_1, \ldots, \mathbf{v}_{M+1}]$ and $\mathbf{c}$ can be used to find the separating hyperplane for the test subject's data as follows:

$$f(\mathbf{X}^{\text{te}}) = \sum_{j=1}^{N} k(\mathbf{X}^{\text{te}}, \mathbf{X}_j^{\text{tr}})\,[\mathbf{v}_1, \ldots, \mathbf{v}_{M+1}] + \mathbf{c} \tag{4.14}$$

As mentioned above, $k(\cdot,\cdot)$ is the kernel function, which is used to measure the similarity between the two input distributions. In their paper, Sangineto et al. [114] experimented with three different kernels. I chose to use their best performing kernel: the Earth Mover's Distance based kernel. The Earth Mover's Distance (EMD) was originally proposed by Rubner et al. in [109]. Computing the EMD-based kernel involves three steps:

1. *Compute "signatures" for each input dataset.* First, each dataset was clustered using k-means. The signatures of each dataset were then set to be the centroids of each cluster and the weights for each cluster were assigned to be the cardinality thereof. The authors fixed k to be 20.
2. *Given the signatures, compute the cost to transform one distribution into the other.* I refer the reader to the EMD paper [109] for details on computing EMD. I used the MATLAB implementation written by Yilmaz [136].
3. *Convert EMD distance into kernel.* Let $D_{EMD}(\mathbf{X}_i, \mathbf{X}_j)$ represent the EMD between $\mathbf{X}_i$ and $\mathbf{X}_j$. Sangineto et al. [114] defined the EMD-based kernel to be the following:
$$k(\mathbf{X}_i, \mathbf{X}_j) = \exp\left(-\rho D_{EMD}\left(\mathbf{X}_i, \mathbf{X}_j\right)\right), \tag{4.15}$$
where ρ is a user-defined parameter.

For more details, I refer the reader to the original TPT paper [114], the M-SVR paper [129], and the EMD paper [109].

4.2 Methods

In this section, I describe some additional, algorithm-specific data processing, the cross validation strategies that I used to select the hyperparameters for each algorithm, and the reasoning used to determine the search range for each hyperparameter. A summary of all algorithms, their associated hyperparameters and the ranges that were searched is given in Table 4.1.

For all algorithms, I used the LAB dataset described in Section 3.1 and the features described in Section 3.4. Algorithm performance was evaluated using leave-one-subject-out (LOSO) cross validation. For each test subject, training data were normalized to have a mean of 0 and a standard deviation of 1, and the test subject's data was adjusted accordingly. Details of model selection are described for each algorithm below.

4.2.1 SVM

SVM uses one hyperparameter, C, which controls the trade-off between maximizing the margin and minimizing the classification error. LIBLINEAR's [37] built in cross-validation function was used to select C for SVM. For most subjects, LIBLINEAR chose C to be 2^{-7}.

4.2.2 KMM

Recall from Eq. 4.2 that KMM solves a quadratic programming problem involving a $n_{\text{tr}} \times n_{\text{tr}}$ kernel matrix $\mathbf{K}$. The complete training dataset has approximately 83k samples (depending

Table 4.1: Hyperparameter ranges for each algorithm

Algorithm	**Hyperparameter**	**Description**	**Range**
SVM	C	Trade-off between maximizing the margin and minimizing the error (Eq. 4.1)	Chosen automatically by LIBLINEAR
KMM	σ	Width of the RBF kernel (Eq. 4.3)	Median distance between all pairs of points
	B	Upper limit for each sample weight s_i (first constraint of Eq. 4.2)	1000
	ε	Maximum value for which mean of weights s can differ from 1 (second constraint of Eq. 4.2)	0.01
	C	SVM trade-off between maximizing the margin and minimizing the weighted error (Eq. 4.4)	$[2^{-13}, \ldots, 2^{-1}]$
STM	σ, B, ε, C	Same as KMM above	Same as KMM above
	λ	Trade-off between the SVM empirical risk and the distribution mismatch 4.5	$[2^{0}, \ldots, 2^{6}]$
TPT	C	Trade-off between maximizing the margin and minimizing the error (Eq. 4.1)	$[2^{-9}, \ldots, 2^{2}]$
	k	Number of centers for k-means when computing the signatures for each dataset (step 1 of computing the EMD-based kernel)	20
	ρ	Width of EMD kernel (Eq. 4.15)	$[2^{-5}, \ldots, 2^{2}]$
	λ	Trade-off between regularizing V and minimizing loss of M-SVR (Eq. 4.9)	$\{1, 10, 100\}$
	ε	Amount of ε-insensitive loss for M-SVR (Eq. 4.10)	$\{10^{-6}, 10^{-3}\}$

on which subject is being excluded as the test subject), and storing a matrix of size $n_{\mathrm{tr}} \times n_{\mathrm{tr}}$ would require approximately 52GB of RAM. Furthermore, as the size of $\mathbf{K}$ increases, so does the time it takes to solve for $\mathbf{s}$. Therefore, I chose to downsample my data to 1/40 of its original size by changing my window shift from 1s to 40s. This amount of downsampling was required for STM (described below) to be able to finish model selection within one day per subject. Although KMM can finish model selection more quickly and could therefore handle more data, the amount of downsampling was fixed between KMM and STM for a fair comparison.

KMM uses four parameters:

- σ controls the width of the RBF kernel. Following the practice of Gretton et al. [45], I chose to use the median distance between all pairs of points in the (training) dataset. Looking at Eq. 4.3, we can see that this choice leads half the values in the exponent to be less than 1/2 and half to be greater.
- B gives an upper bound to the weights s_i. The limit used by Gretton et al. [45] was 1000 and results presented in this chapter used the same value. I experimented with several values and, consistent with findings from Gretton et al., I did not see significant variation in performance and I found that the maximal s_i value rarely exceeded a few hundred.
- KMM attempts to assign weights s_i such that the average weight is 1. ε specifies the maximal deviation of the average from 1. Interestingly, Gretton et al. [45] suggest that a good choice of ε should be $O(B/\sqrt{(n_{\mathrm{tr}})})$, which is approximately 32 for this dataset. A value that large would make the constraint of the maximal deviation of the average from 1 to be nearly meaningless. Therefore, I chose to use a small value of 0.01.
- After KMM has solved for the weights s_i, I solve a weighted SVM to learn the separating hyperplane, which uses the hyperparameter C. Given that the optimal value of C for the unweighted SVM was typically 2^{-7}, I chose to perform cross validation across the range of 2^{-13} to 2^{-1}.

With three of the four parameters set to fixed values, model selection only needed to be performed on the hyperparameter C. Note that, because KMM relies on the distribution of the test (or validation) data, the standard practice of choosing validation sets – *i.e.*, random sampling of the training set – may lead to inaccurate parameter estimates. For validation sets to better represent the test set, they should each come from a single subject. Therefore, each choice for C was validated using an inner loop over training subjects: Data from both hands of a test subject were partitioned off as test data. Data from both hands of one of the training subjects were then partitioned off as a validation set. Note that, when inputting the unlabeled $\mathbf{X}^{\mathrm{te}}$ data into KMM, data from each hand were input separately, rather than combining them into a single $\mathbf{X}^{\mathrm{te}}$. All values of C were validated on data from each hand of the validation subject, and this process was repeated for each subject in the training set. The C with the best average performance (where performance was measured either using accuracy or AUC) was then chosen for training KMM. As with the validation subjects, KMM was run twice, inputting data from each hand of the test subject separately as $\mathbf{X}^{\mathrm{te}}$.

4.2.3 STM

As described above for KMM, data was downsampled forty fold such that the kernel matrix could fit in memory and model selection could complete in a timely manner.

With respect to hyperparameters, STM employs all the same ones as KMM with an additional λ parameter to control the trade-off between the weighted SVM empirical risk and the distribution mismatch between the training and test sets. For all parameters shared with KMM, I chose to use the same values as KMM. To choose the search range for λ, the two parts of the objective function were analyzed and a range that would balance the magnitudes was selected. This range was 2^0 to 2^6.

The model selection procedure for STM was the same as described above for KMM. With 13 values of C and 7 values of λ, a total of 91 possible pairs of parameters were searched, which took approximately 1.5 hours. Model selection for each test subject involves evaluating all parameter pairs across all validation sets (9 validation subjects $\times$ 2 hands = 18 validation sets), and therefore takes approximately 25 hours per subject.

4.2.4 TPT

TPT uses five different hyperparameters:

- Given that TPT is based on an SVM, we again have the hyperparameter C. However, because TPT trains an SVM on each training subject, rather than on the entire dataset at once, the number of samples input into the SVM is much less, which means that the range chosen for C should differ from that used for the generic SVM. Looking at the performance for various choices of C, I found that the optimal value when training on a single subject tended to be approximately within 2^{-8} to 2^1. Therefore, I chose to search the range of 2^9 to 2^2.
- TPT uses k-means to cluster the data from each subject and then uses the cluster centers to define a "signature" for each subject. Sangiento et al.[114] fixed the value of k to be 20, and I did the same for this thesis.
- From Eq. 4.15, we can see that ρ defines the kernel width. For KMM, I chose the kernel width to be the median distance, leading the median value within the exponent to be 1/2. The median value for D_{EMD} was approximately 2.5. Taking a similar approach to choosing ρ would lead to $\rho \approx 2^{-2}$. Therefore, I chose to search the range 2^{-5} to 2^2.
- λ defines the trade-off between regularizing $\mathbf{B}$ and minimizing the ε-insensitive loss in M-SVR. I searched the same set of values that was used in [129]: $\{1, 10, 100\}$.
- ε denotes the level of ε-insensitive loss in M-SVR. I used the same set of values described in [129]: $\{10^{-6}, 10^{-3}\}$.

To choose C, the performance of each choice of C was evaluated via five-fold cross-validation on data from each hand of each training subject. The matrix $\boldsymbol{\Theta}$ was built by using the best performing C to train an SVM on each hand of each subject. As described above, the number of groups k for computing signatures of each dataset was fixed. The remaining parameters ρ, λ and ε were selected using an inner loop on validation subjects, as described above for SVM and KMM.

4.3 Results and discussion

All four algorithms were applied to the LAB dataset (see Chapter 3.1). Tables 4.2 and 4.3 show results of the four algorithms when using accuracy and AUC, respectively, during model selection of the hyperparameters. Currently, LIBLINEAR only supports using accuracy for model selection. Therefore, SVM results are identical in both tables. For each model, accuracy and AUC on the test dataset are reported. On average, both tables show that results of the personalized models (KMM, STM and TPT) do not differ significantly from the generic SVM.

The performance of TPT is surprisingly low. Tables 4.2 and 4.3 show that using AUC for model selection improves performance of TPT over using accuracy. However, TPT is still worse, on average, than the generic SVM in accuracy and AUC regardless of which performance metric is used during model selection. In their paper, Sangineto et al. [114] demonstrated a 0.07 improvement (averaged over six different facial action units) in AUC over a generic SVM on the Extended Cohn-Kanade (CK+) dataset. The CK+ dataset contains 593 videos from 123 subjects. There are 1 to 11 videos per subject and 4 to 71 frames per video. The authors extracted a 51 dimensional feature vector from each frame. In comparison, this dataset uses a 21 dimensional feature vector from 10 subjects. Experiments by Sangineto et al. [114] indicated that performance of TPT was variable with fewer subjects and that TPT would sometimes perform worse than a generic SVM depending on the subset of subjects chosen. Differences in the datasets lead to several possible explanations for the poor performance of TPT:

- Having fewer subjects in this dataset may make it more difficult to learn a good regression function.
- A larger number of samples per subject may require a larger value of k to accurately represent the distribution with k-means clustering.
- The dataset and/or feature vectors may require different or larger sets of parameters to be searched.
- It is possible that the variability between subjects in the feature space is higher in this dataset, or that the feature space is less well-covered, which could lead to difficulty in learning the regression function.

The Tables 4.2 and 4.3 also show that STM is more sensitive to which performance metric is used during model selection (greater change in performance between tables), while average results for KMM seem quite robust to the model selection process. In contrast to the other algorithms, where accuracy and AUC are loosely correlated, STM will optimize one over the other: accuracy decreases and AUC increases from Table 4.2 to Table 4.3. It is possible that for STM, some of the models trained for validation subjects have high accuracy and low AUC (or vice versa), which could lead to very different choices of parameters depending on the performance metric used. In contrast, KMM uses only one parameter, which limits the

Table 4.2: Results when using accuracy to measure performance during model selection. Bold indicates best accuracy and best AUC for each test dataset.

		Accuracy				**AUC**			
Subject	**Hand**	SVM	KMM	STM	TPT	SVM	KMM	STM	TPT
2	L	72.4	**90.4**	87.2	74.6	0.85	**0.89**	0.86	0.86
	R	79.5	82.4	82.4	**85.3**	0.85	0.89	0.89	**0.91**
3	L	61.3	**62.6**	58.7	50.1	0.77	0.75	0.73	**0.78**
	R	62.3	65.2	**75.1**	48.0	0.80	0.79	0.73	**0.83**
4	L	**83.6**	81.1	78.5	77.1	**0.90**	0.88	0.88	0.89
	R	**88.6**	88.3	85.9	68.8	0.91	0.91	**0.92**	0.63
5	L	67.7	68.1	**74.3**	64.5	0.77	0.79	**0.79**	0.76
	R	64.2	63.4	**70.5**	63.5	0.77	**0.79**	0.78	0.74
7	L	75.8	**76.3**	74.4	75.0	**0.81**	0.80	0.75	0.80
	R	**83.8**	83.3	82.2	82.2	0.85	**0.85**	0.84	0.73
8	L	**91.7**	91.4	91.5	89.8	0.82	0.82	0.75	**0.83**
	R	81.0	81.2	81.4	**81.8**	0.88	0.88	0.82	**0.88**
9	L	78.8	78.8	78.5	**79.3**	0.64	**0.72**	0.70	0.65
	R	87.5	90.2	**90.9**	90.0	0.66	**0.69**	0.59	0.65
10	L	85.2	86.2	84.9	**86.7**	0.72	0.80	**0.81**	0.72
	R	84.6	85.8	**87.3**	87.3	0.78	0.82	**0.85**	0.78
11	L	**86.2**	85.2	73.4	81.1	**0.92**	0.91	0.87	0.88
	R	83.7	84.6	**86.5**	72.8	0.89	0.90	**0.90**	0.69
12	L	86.8	87.0	**88.2**	80.8	**0.87**	0.75	0.76	0.85
	R	82.9	82.5	**97.5**	59.9	0.79	0.77	**0.79**	0.48
Average		79.4	80.7	**81.5**	74.9	0.81	**0.82**	0.80	0.77

Table 4.3: Results when using AUC to measure performance during model selection. Bold indicates best accuracy and best AUC for each test dataset.

		Accuracy				**AUC**			
Subject	**Hand**	SVM	KMM	STM	TPT	SVM	KMM	STM	TPT
2	L	72.4	**89.5**	87.9	82.6	0.85	**0.88**	0.83	0.87
	R	79.5	**82.4**	80.5	81.9	0.85	**0.89**	0.88	0.88
3	L	61.3	63.4	**64.3**	53.1	0.77	0.75	0.75	**0.78**
	R	62.3	69.9	**72.6**	49.3	0.80	0.79	0.73	**0.82**
4	L	**83.6**	79.9	78.7	79.2	**0.90**	0.87	0.87	0.89
	R	**88.6**	88.4	87.8	73.2	0.91	0.90	**0.91**	0.76
5	L	67.7	72.2	**72.9**	69.5	0.77	**0.79**	0.79	0.77
	R	64.2	67.4	**69.2**	67.1	0.77	0.78	**0.78**	0.73
7	L	75.8	75.7	**78.2**	75.9	0.81	0.82	**0.84**	0.80
	R	83.8	83.6	**84.3**	83.0	0.85	0.88	**0.89**	0.83
8	L	**91.7**	91.6	79.3	90.0	0.82	**0.83**	0.78	0.83
	R	81.0	79.6	78.8	**81.7**	0.88	0.86	0.85	**0.88**
9	L	78.8	79.0	78.7	**79.5**	0.64	0.72	**0.73**	0.65
	R	87.5	**88.7**	87.9	88.2	0.66	0.67	**0.68**	0.67
10	L	85.2	86.4	84.2	**86.9**	0.72	0.76	**0.83**	0.73
	R	84.6	86.8	**87.7**	86.6	0.78	0.81	**0.88**	0.78
11	L	**86.2**	85.0	85.9	70.6	0.92	0.90	**0.93**	0.90
	R	83.7	84.1	**84.8**	79.1	0.89	0.89	**0.90**	0.86
12	L	86.8	86.4	77.7	**96.3**	0.87	0.74	0.84	**0.92**
	R	82.9	82.4	33.3	**93.7**	0.79	0.79	0.75	**0.83**
Average		79.4	**81.1**	77.7	78.4	0.81	**0.82**	**0.82**	0.81

Table 4.4: Results for the highest performing parameter set. Bold indicates best accuracy and best AUC for each test dataset.

		Accuracy				**AUC**			
Subject	**Hand**	SVM	KMM	STM	TPT	SVM	KMM	STM	TPT
2	L	72.7	90.5	**90.8**	89.5	0.85	0.89	**0.89**	0.89
	R	79.7	82.8	83.9	**85.3**	0.85	0.89	0.91	**0.91**
3	L	61.4	64.7	**70.9**	67.6	0.77	0.77	0.75	**0.78**
	R	62.3	72.6	**78.7**	66.6	0.80	0.79	0.79	**0.83**
4	L	**83.8**	82.5	80.4	81.0	**0.90**	0.89	0.89	0.89
	R	**89.5**	89.0	88.7	86.5	0.92	0.92	**0.93**	0.90
5	L	67.9	72.2	**75.3**	72.9	0.77	0.79	0.79	**0.82**
	R	64.4	67.4	**70.8**	67.1	0.77	**0.79**	0.79	0.79
7	L	75.8	76.4	**80.5**	76.3	0.81	0.83	**0.86**	0.81
	R	83.8	84.1	**84.9**	83.0	0.85	0.88	**0.89**	0.88
8	L	**91.9**	91.8	91.7	90.4	0.82	0.83	0.81	**0.84**
	R	81.1	81.2	81.5	**81.8**	0.88	0.88	0.87	**0.88**
9	L	78.8	79.4	79.4	**79.5**	0.64	0.72	0.74	**0.74**
	R	87.5	90.3	91.9	**92.2**	0.66	**0.70**	0.68	0.68
10	L	85.3	86.4	86.0	**87.0**	0.72	0.82	**0.83**	0.74
	R	84.8	86.9	**88.3**	87.4	0.78	0.85	**0.88**	0.78
11	L	87.1	**88.4**	88.3	81.8	0.93	0.94	**0.94**	0.91
	R	83.9	85.5	**87.5**	84.6	0.89	0.91	**0.91**	0.87
12	L	87.8	87.0	96.2	**96.3**	0.87	0.80	0.84	**0.92**
	R	85.0	82.5	**98.8**	94.1	0.79	0.84	**0.88**	0.85
Average		79.7	82.1	**84.7**	82.5	0.81	**0.84**	**0.84**	**0.84**

effect of parameter choice on performance, and TPT learns a regression from distributions to person-specific "ideal" classifiers, which are less likely to have divergent accuracy and AUC values.

Model selection is non-trivial and performance can be sensitive to the choice of parameters. To compare algorithms independent of the model selection process, Table 4.4 shows results for the highest performing parameter set for each test set. We can see that STM has the greatest potential to improve accuracy when the best parameters are chosen, even though the number of hyperparameters is equal to that of TPT. In fact, the number of *free* hyperparameters for STM is only two (C and λ), compared to four free parameters of TPT (C, ρ, λ and ε). Interestingly, the average AUC values are identical across all three personalization algorithms.

4.4 Conclusions and future work

In this chapter, I compared PD tremor detection performance of a generic SVM to three methods of personalizing an SVM given unlabeled data from the test subject. Results show that all three personalization methods have the potential to achieve higher performance than the generic SVM method given appropriate parameters. However, when using model selection to choose hyperparameters, TPT tends to do worse than a generic SVM (possibly due to differences between our dataset and theirs), and STM is sensitive to the method of model selection (possibly because some of the models learned during model selection have divergent values for accuracy and AUC). In the next chapter, the behavior of STM is analyzed in more detail on a dataset for human activity recognition and several synthetic datasets.

As shown in Table 4.1, each of the personalization algorithms requires the user to set a number of hyperparameters. Choosing an appropriate search range for each parameter is non-trivial, and running an inner loop across validation subjects to select values for the hyperparameters is computationally expensive. The parameters suggested by Gretton et al. [45] seem to work well for KMM (leaving only C to be selected). Meanwhile, it is possible that the ranges suggested for the M-SVR part of TPT were too restrictive.

Of the three personalization algorithms – KMM, STM and TPT – KMM offers better performance than TPT (with model selection), is faster than STM, and is more robust to the model selection procedure. It would be interesting to explore techniques for speeding up model selection or choosing better parameters, which might enable the personalization algorithms to achieve their full potential. For example, one could try to reason about appropriate parameter values or ranges given a dataset. Alternatively, making the parameters agnostic to the size of the dataset could enable model selection to be performed on smaller datasets, which require less training time. Another interesting avenue of research is to try to intelligently exclude some of the parameter sets while searching the parameter space.

Chapter 5

Analyzing the Selective Transfer Machine

Previous work on STM in facial action unit detection demonstrated improvement from STM over KMM and SVM on multiple datasets. Results from Chu et al. [25] indicated that the performance of KMM tended to be on par with, or worse than, a generic SVM, while the performance of STM was often better. In Sangineto et al. [114], TPT was also able to show improvement in the task of facial recognition. The results on the PD dataset in Chapter 4.3 – that KMM and STM performed similarly, and the TPT performed *worse* than the generic SVM – were surprising. It was possible that differences in the application domain, feature space, data dimensionality, or the datasets were the cause of these unexpected findings. This chapter therefore presents in-depth experiments of STM conducted on a human activity recognition dataset and on synthetic datasets to better understand its behavior and performance.

This chapter is organized as follows. Because human activity recognition is often a multiclass problem, Section 5.1 presents multiclass extensions to SVM and STM. The chosen human activity recognition dataset, REALDISP [12, 13], is described in Section 5.2, and preliminary results of these multiclass extensions on REALDISP are presented in Section 5.3. Interestingly, relative performance between SVM, KMM, and STM on REALDISP were similar to those reported on the PD data in Chapter 4.3. Section 5.4 describes experiments on synthetic datasets for further analysis. On these synthetic datasets, STM still struggled to outperform KMM. Therefore, Section 5.5 describes two modifications to the STM algorithm that were designed to help improve its performance on the synthetic datasets. Section 5.5 also presents the results on some of the synthetic datasets and the REALDISP dataset. While these modifications demonstrated improvement in the synthetic datasets, they were still unable to outperform SVM and KMM on REALDISP. In Section 5.6, the STM objective function is analyzed and global optimization is explored to try to understand why performance of STM is not as high as expected. It is found that, on some datasets, minimizing the STM objective function does not increase accuracy. While it is unclear why STM was able to perform so well in the facial action unit domain, results from this chapter suggest that the STM objective function is not appropriate for the domain of activity recognition from accelerometer data.

5.1 Overview of multiclass algorithms

Human activity recognition is typically a multiclass problem. This section therefore describes general methods for applying binary classifiers to multiclass problems. It also describes several different versions of multiclass SVM, either by applying a binary SVM to a multiclass problem, or by reformulating the SVM objective function to be inherently multiclass. The concepts of multiclass SVM are then applied to STM to develop a multiclass version STM.

5.1.1 Converting binary classifiers to multiclass

Many classification algorithms can naturally handle multiclass scenarios, such as k-NN or decision trees. For binary classifiers, however, two main methods exist for applying them to multiclass problems:

- *One-versus-all (OVA)* – In the OVA approach, n classifiers are trained – one for each of the n classes. For the ith classifier, all points labeled as class i are selected as positive samples and the rest are considered negative samples. A standard binary classifier is then trained. During testing, all test samples are classified by all n classifiers. Test samples are labeled with the class of the classifier that outputs the largest score.
- *One-versus-one (OVO)* – In the OVO approach, $\frac{n(n-1)}{2}$ binary classifiers are trained – one for each pair of the n classes. During testing, each classifier votes on the class, and the class that receives the most votes is chosen.

5.1.2 Multiclass SVM

The SVM algorithm is inherently a binary classifier. As discussed in above, OVA and OVO are two common methods for adapting binary classifiers to multiclass problems. In the case of SVM, however, several formulations that solve for all classes at once have been proposed. These methods are referred to as *single machine* algorithms.

In general, all single machine formulations follow the same basic structure. Suppose we have m classes and m decision boundaries defined by $\mathbf{w}_j$, with $j = 1, \ldots, m$. Let $f_j(\mathbf{x}_i)$ be the decision function of the jth class evaluated on the sample $\mathbf{x}_i$. In the linear case, $f_j(\mathbf{x}_i) = \mathbf{w}_j^\top \mathbf{x}_i + b_j$. Then, the single machine SVM objective function is given by

$$\min_{\mathbf{w}_j} \frac{1}{2} \sum_{j=1}^{m} \mathbf{w}_j^\top \mathbf{w}_j + C \sum_{i=1}^{n_{\text{tr}}} L(y_i, f_1(\mathbf{x}_i), \cdots, f_m(\mathbf{x}_i)). \tag{5.1}$$

Note how Eq. 5.1 is very similar to Eq. 4.1, except that we are now summing the norms of each of the m $\mathbf{w}_j$ vectors. The various single machine approaches only differ in their loss function $L(y_i, f_1(\mathbf{x}_i), \cdots, f_m(\mathbf{x}_i))$. Three formulations have been extensively studied in the literature:

- Weston and Watkins [133]:

$$L_{WW}(y_i, f_1(\mathbf{x}_i), \cdots, f_m(\mathbf{x}_i)) = \sum_{j \neq y_i} \max\left\{2 - f_{y_i}(\mathbf{x}_i) + f_j(\mathbf{x}_i), 0\right\}. \tag{5.2}$$

  This loss function implies that the score from the true class should be greater than that from any other by a margin of 2.

- Crammer and Singer [30]:

$$L_{CS}(y_i, f_1(\mathbf{x}_i), \cdots, f_m(\mathbf{x}_i)) = \max\left\{1 - f_{y_i}(\mathbf{x}_i) + \max_{j \neq y_i} f_j(\mathbf{x}_i),\ 0\right\}.$$

  This loss function implies that the score from the true class should be greater than the highest score from any false class by a margin of 1.

- Lee, Lin and Wahba [79]:

$$L_{LLW}(y_i, f_1(\mathbf{x}_i), \cdots, f_m(\mathbf{x}_i)) = \sum_{j \neq y_i} \max\left\{f_j(\mathbf{x}_i) + \frac{1}{m-1},\ 0\right\},$$

  where m is the number of classes. This loss function implies that the score of a false class should be less than zero by a margin of $\frac{1}{m-1}$.

Dogan et al. provide a nice comparison of the three methods in terms of training time and accuracy [35].

Several comparisons of the three multiclass SVM approaches (OVA, OVO, and single machine) have been published, although the most thorough is that of Hsu and Lin [60]. The general consensus is that the OVA or OVO methods are simpler to implement and do not differ significantly in performance from the single machine approaches.

5.1.3 Multiclass Selective Transfer Machine

This section presents the multiclass extensions to the original STM algorithm. As with all binary classifiers, we can apply a binary STM to multiclass problems by using OVA or OVO. In this section, we also derive a single machine multiclass STM, which combines the formulations of STM and multiclass SVM using the Weston and Watkins loss function. Hence, it is called WW-STM. Combining the formulations simply involves replacing the empirical risk of the penalized binary SVM term, R_f, in Eq. (4.5) with the penalized multiclass SVM:

$$R_{\mathbf{w}_j}(\mathcal{D}^{\mathrm{tr}}, \mathbf{s}) = \min_{\mathbf{w}_j, b_j, \mathbf{s}} \frac{1}{2}\sum_{j=1}^{m} \mathbf{w}_j^\top \mathbf{w}_j + \frac{C}{2}\sum_{i=1}^{n_{\mathrm{tr}}} s_i\, L(y_i, f(\mathbf{x}_i)) f_1(\mathbf{x}_i), \cdots, f_m(\mathbf{x}_i).$$

In general, $L(y_i, f_1(\mathbf{x}_i), \cdots, f_m(\mathbf{x}_i))$ could be any of the single machine SVM loss functions as described in Section 5.1. However, that of Weston and Watkins was chosen for its simplicity, particularly when optimizing in the primal. Furthermore, to ensure differentiability of our objective function, the $L2$ version of Eq. (5.2) was used:

$$L_{WW}(f_1(\mathbf{x}_i), \cdots, f_m(\mathbf{x}_i)) = \sum_{j \neq y_i} \max\{2 - f_{y_i}(\mathbf{x}_i) + f_j(\mathbf{x}_i), 0\}^2 .$$

Note how the risk term, $R_{\mathbf{w}_j}$, is minimized over the m separate $\mathbf{w}$ vectors and b scalars. In order to solve for all classes simultaneously, we now proceed to reformulate it to be over a single $\mathbf{W}$ matrix, where

$$\mathbf{W} = \begin{bmatrix} | & & | \\ \mathbf{w}_1 & \cdots & \mathbf{w}_m \\ | & & | \\ b_1 & \cdots & b_m \end{bmatrix}.$$

Let $\mathbf{I}^0$ be a $(d+1) \times (d+1)$ matrix with ones on the diagonal, a zero in the last index of the diagonal, and zeros everywhere else. Let $\mathbf{1}_j$ be a selector vector of length m with 1 in the jth index and zeros everywhere else. Then, $\mathbf{w}_j = \mathbf{I}^0\mathbf{W}\mathbf{1}_j$. Let $\hat{\mathbf{x}}_i$ be the concatenation of $\mathbf{x}_i$ and 1: $\hat{\mathbf{x}}_i{}^\top = \left[\mathbf{x}_i^\top \; 1\right]$.

The multiclass SVM with weighted loss can now be reformulated as follows:

$$\mathcal{R}_{\mathbf{W}}(\mathcal{D}^{\mathrm{tr}}, \mathbf{s}) = \min_{\mathbf{W},\mathbf{s}} \frac{1}{2}\sum_{j=1}^{m}(\mathbf{I}^0\mathbf{W}\mathbf{1}_j)^\top(\mathbf{I}^0\mathbf{W}\mathbf{1}_j) + \frac{C}{2}\sum_{i=1}^{n_{\mathrm{tr}}} s_i \, L_{WW}\left(y_i, \mathbf{W}, \hat{\mathbf{x}}_i\right),$$

where

$$L_{WW}\left(y_i, \mathbf{W}, \hat{\mathbf{x}}_i\right) = \sum_{j \neq y_i} \max\left\{(\mathbf{1}_j - \mathbf{1}_{y_i})^\top \mathbf{W}^\top \hat{\mathbf{x}}_i + 2, 0\right\}^2 .$$

Holding $\mathbf{s}$ constant, we can solve for $\mathbf{W}$ in the primal by following the method suggested by Chapelle [22]. Because the $\boldsymbol{\Omega}$ part of the multiclass STM does not contain $\mathbf{W}$, we only need to compute the gradient of $R_{\mathbf{W}}$:

$$\nabla = \sum_{j=1}^{m} \mathbf{I}^0\mathbf{W}\mathbf{1}_j\mathbf{1}_j^\top + C\sum_{i=1}^{n_{\mathrm{tr}}} s_i \frac{\partial L_{WW}}{\partial \mathbf{W}},$$

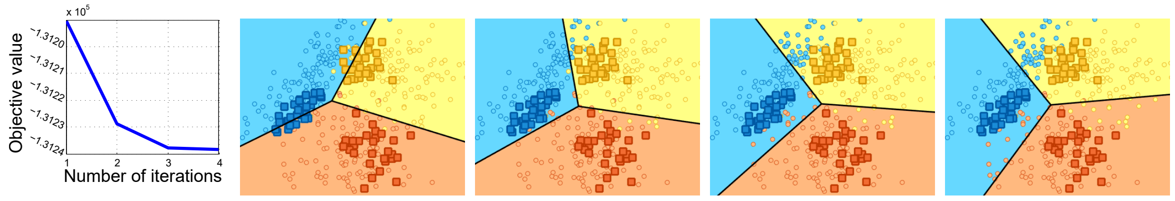
where

$$\frac{\partial L_{WW}}{\partial \mathbf{W}} = \sum_{\mathrm{sv}_j} \left((\mathbf{1}_j - \mathbf{1}_{y_i})^\top \mathbf{W}^\top \hat{\mathbf{x}}_i + 2\right) \hat{\mathbf{x}}_i (\mathbf{1}_j - \mathbf{1}_{y_i})^\top .$$

Here, Chapelle's strategy [22] for differentiating the max term was used, where sv_j represent the values j for which, given $\mathbf{x}_i$, $(\mathbf{1}_j - \mathbf{1}_{y_i})^\top \mathbf{W}^\top \hat{\mathbf{x}}_i + 2 > 0$.

Using the gradient, we can use line search to solve for $\mathbf{W}$. We solve for $\mathbf{s}$ using quadratic programming, as discussed above in Section 4.1.3. The only difference is that the SVM loss term corresponding to each s_i is the Weston and Watkins loss, *i.e.* $\ell_i = L_{WW}\left(y_i, \mathbf{W}, \hat{\mathbf{x}}_i\right)$. As in STM, WW-STM is solved by iteratively solving for $\mathbf{W}$ and $\mathbf{s}$.

Figure 5.1 illustrates the process of STM. STM is initialized to SVM, as shown in iteration # 1. Then, samples are reweighted according to the training loss and distribution mismatch. Performance on the test set improves as STM iteratively approaches the ideal classifier, which is an SVM trained on the test points.



Convergence curve | Iteration #1 (Te%=85.6,Tr%=93.7) | Iteration #2 (Te%=100,Tr%=88.7) | Iteration #4 (Te%=100,Tr%=77.7) | Ideal classifier (Te%=100,Tr%=73.3)

Figure 5.1: (Left to right) Convergence curve of the multiclass STM, separation hyperplane at iterations #1, #2, and #4, and the ideal classifier (SVM trained on the test samples). Circles (◯) and squares (□) indicate training and test samples, respectively. Te% and Tr% indicate the accuracy on test and training data. Note how, as the STM iteration proceeds, the STM hyperplane approaches the ideal hyperplane for the test data and the performance on the training data decreases.

5.2 REALDISP dataset

After comparing many datasets in the activity recognition domain, the REALDISP dataset [12, 13] for wearable activity recognition was chosen due to its relatively large number of subjects and diverse number of classes. This dataset consists of 17 subjects performing 33 activities while wearing nine inertial measurement units collecting at 50 Hz. To better simulate the scenario of activity classification on a smart watch or fitness tracker, only data from the right wrist sensor was used. Following [12], the mean and standard deviation of the accelerometer signal were computed on each axis over non-overlapping, six-second windows, resulting in six features.

Figure 5.2 depicts all 33 classes and the number of samples (feature vectors) per subject, per activity. Figure 5.3 (left) gives the distribution of the 33 activity classes across all subjects. Note that some subjects are missing classes and the overall distribution of samples across the classes is unbalanced.

5.3 Preliminary results on REALDISP

Table 5.1 shows results of multiclass SVM, KMM and STM using the OVA, and WW paradigms. Interestingly, we can see that OVO consistently outperforms the OVA and WW methods. This result is consistent with some of the previous findings in the literature [60]. We also note a performance drop in OVO-STM versus OVO-SVM. It is possible that, in the OVO case, proper sample reweighting is particularly difficult. This difficulty may occur because, when training each of the OVO classifiers, STM is given training data that contains only two classes, and tries to match this distribution to the (unlabeled) test data, which contains many classes. However, in the case of OVA and WW, we can see that the methods that consider importance reweighting (STM and KMM) offer a significant performance im-

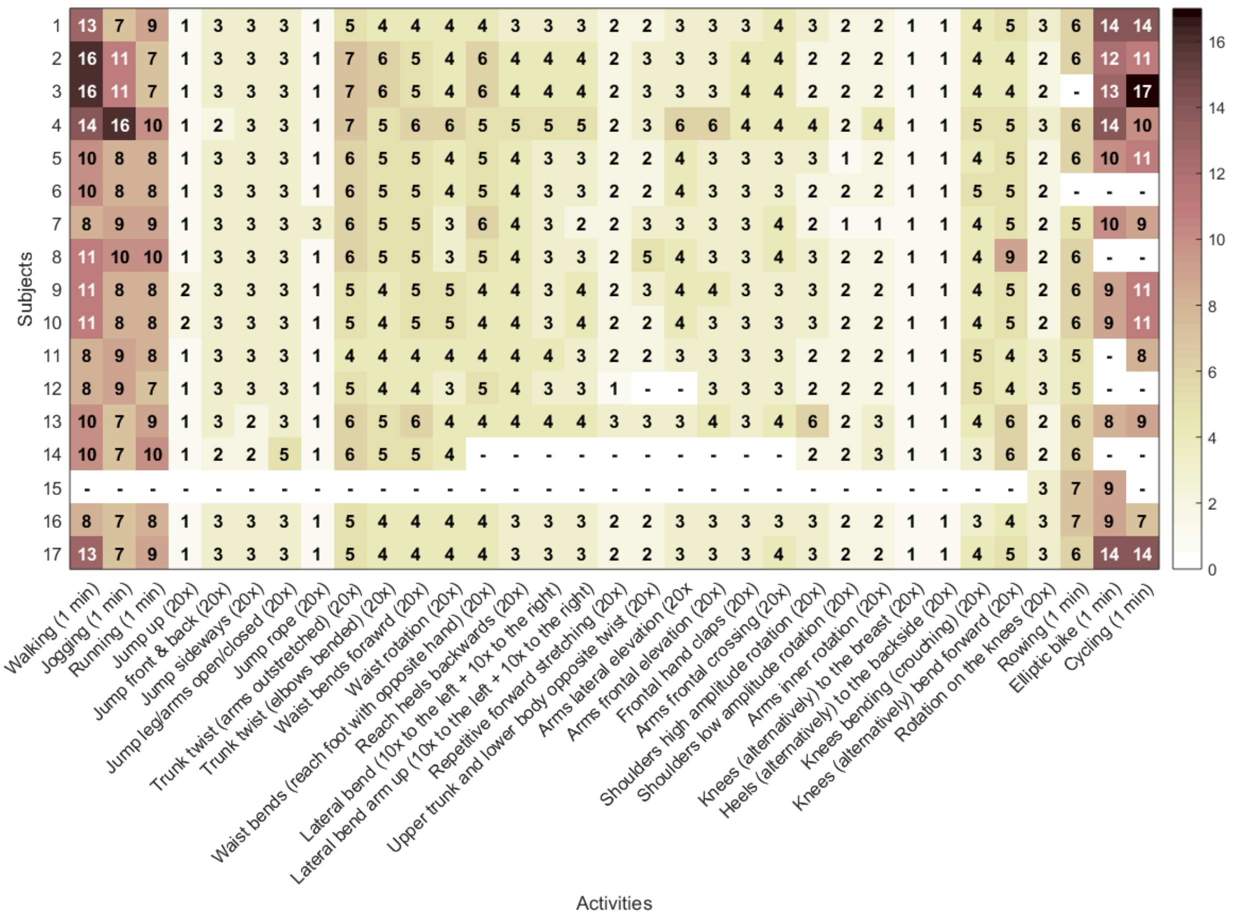
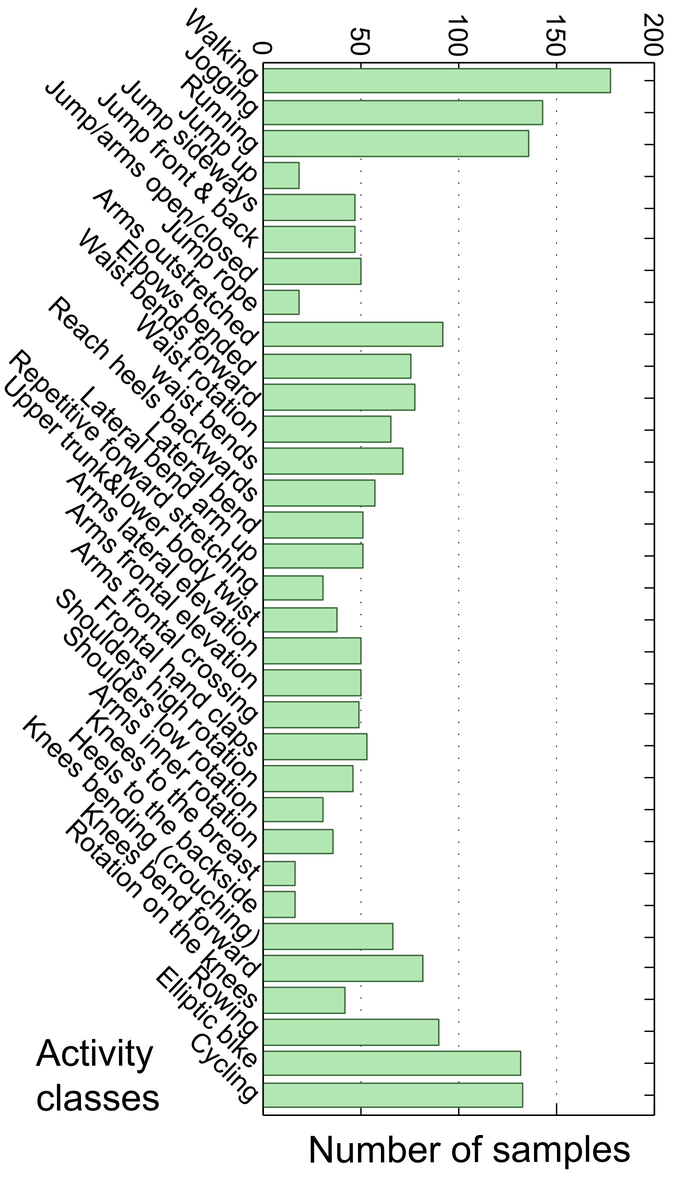


Figure 5.2: Number of samples in REALDISP for each subject and activity

provement over SVM (approximately 10 points in the F1-score). We note that WW-STM offers a slight performance improvement over WW-KMM in accuracy, although there is a slight decrease in the F1-score between WW-KMM and WW-STM. This decrease may be due to an imbalance of the class sizes. Some classes in REALDISP have ten times as many samples as others, as can be seen in Figure 5.3 (left). Because the average F1-score over all classes was computed, improving classification on one or two samples for a small class can dramatically affect the averaged F1-score. Figure 5.3 (right) gives the full breakdown of F1-score by class for WW-SVM, KMM and STM. We can see that STM generally has comparable or increased performance on the larger classes, but occasionally has reduced performance on some of the smaller classes. This behavior is likely an effect of the importance reweighting. Checking the partial confusion matrices in Figure 5.4 of WW-SVM (top) and WW-STM (bottom), we can see that the confused classes are reasonable, particularly for classifying from only the right wrist sensor: class 2 (“jogging”) is confused with class 3 (“running”); and classes 4, 5 and 6 (“jump up,” “jump sideways,” and “jump front & back”



Class	WW-SVM	WW-KMM	WW-STM
1	54.7	**71.0**	68.4
2	55.7	55.2	**60.6**
3	52.9	53.1	**54.2**
4	0.0	**25.0**	9.3
5	**43.1**	21.6	27.9
6	19.4	23.9	**25.7**
7	73.6	**78.1**	**78.1**
8	**38.7**	32.0	12.0
9	72.6	**79.2**	77.7
10	61.4	**69.6**	64.0
11	42.2	37.0	**52.3**
12	43.9	**56.5**	51.2
13	**66.2**	55.1	59.0
14	**63.2**	44.6	60.4
15	29.7	36.3	**40.0**
16	2.9	**39.2**	34.4
17	12.5	33.9	**45.4**
18	4.5	**43.3**	26.9
19	46.3	**67.3**	57.7
20	**73.6**	63.7	60.6
21	80.4	**85.3**	78.1
22	62.6	68.6	**70.2**
23	40.0	47.6	**50.5**
24	0.0	**68.7**	63.3
25	55.8	**71.2**	62.3
26	0.0	5.7	**9.7**
27	0.0	**43.2**	25.6
28	4.4	14.8	**19.8**
29	19.8	47.3	**54.5**
30	50.6	**75.6**	74.0
31	82.0	**88.6**	87.9
32	78.1	**85.4**	83.8
33	**89.8**	83.9	88.4
Avg	43.0	**53.7**	52.5

Figure 5.3: Performance on REALDISP by activity class. Left: The number of samples of each activity class in the REALDISP dataset. Right: The F1-score for each class from WW-SVM, WW-KMM and WW-STM.

Table 5.1: 2-fold model selection performance comparison

	Accuracy (%)			F1-score		
Methods	SVM	KMM	STM	SVM	KMM	STM
OVA	55.6	58.4	57.3	43.0	52.1	52.8
OVO	67.7	63.3	64.8	60.9	58.0	56.6
WW	56.3	59.4	59.9	43.1	53.7	52.6

Table 5.2: Gold-standard performance comparison

	Accuracy (%)			F1-score		
Methods	SVM	KMM	STM	SVM	KMM	STM
OVA	–	–	–	–	–	–
OVO	–	–	–	–	–	–
WW	59.0	60.4	64.5	49.2	54.9	56.5

Note: Results for OVA and OVO are omitted due to computational cost.

respectively) are confused with each other.

Although the performance of OVA- and WW-STM were similar, the training times were not. It was found found that training OVA-STM took approximately 10 times as long as WW-STM, which was surprising because previous work has generally found single machine classifiers to be slightly slower than OVA or OVO [60]. Furthermore, our implementation of WW-STM used first-order gradient descent to solve for $\mathbf{W}$, whereas the OVA-STM implementation uses a second-order method, which should make it faster. It is likely that OVA-STM is slow due to the KMM step, where a kernel matrix of size $n_{\text{tr}} \times n_{\text{tr}}$ is created. WW-STM essentially reduces the cost of this step by the number of classes because all classes are solved for at once, with the consequence of adding complexity to the solving of $\mathbf{W}$. Creating the kernel matrix was not a problem in OVO-STM because the size of the training set is dramatically reduced given that only two classes are considered at once. In conclusion, in the case of many classes, OVA-STM should be avoided in favor of WW-STM.

We note that the performance increase offered by WW-STM over WW-SVM is not particularly dramatic on the REALDISP dataset. One cause may be a choice of parameters. Recall that STM jointly optimizes a decision boundary and the distribution mismatch. Hence, the weights on the training points and the size of the margin are dependent on the distribution of the test set. Therefore, if the distribution of the validation and test set do not match, the validation set may not accurately reflect performance on the test set. That is, the parameters that perform best on the validation set may not perform best on the test set.

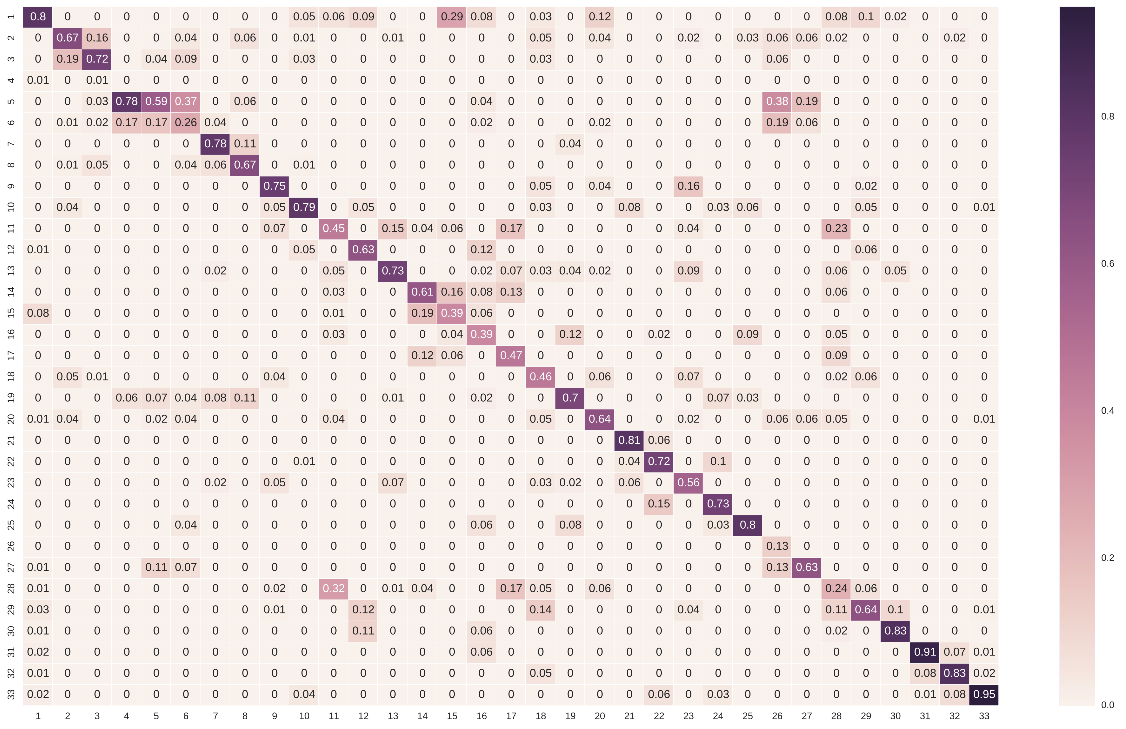
To check performance of STM independent of this model selection issue, gold standard results are reported in Table 5.2. Gold standard is defined as the performance obtained by using the best possible parameter set, which is found by evaluating the performance of every classifier on the test set. Note that the gold standard results for OVA and OVO were excluded because the results for every possible combination of parameters would need to be computed, which would be computationally too expensive. For a given algorithm, the number of parameter combinations is given by $\#\text{classes}^{\#\text{parameter values}}$. Therefore, the number of combinations ranges from 33^8 for OVA-SVM up to $\binom{32}{2}^{64}$ for OVO-STM. Gold standard results on the WW method show a more significant performance improvement from STM over the other algorithms, with an accuracy of 64.5% compared to 60.4% and 59.0% obtained by SVM and KMM, respectively. These findings are consistent with those from Chapter 4.3 on the PD dataset, where STM obtained an accuracy of 84.7% compared to 82.1% and 79.7% from SVM and KMM, respectively (Table 4.4). Using a better model selection paradigm could therefore help increase performance of STM over SVM.

STM has previously been shown to give improvements of approximately 5-10 points in AUC over SVM and KMM in facial action unit detection [24]. It was therefore surprising to see the more modest improvements on the PD and REALDISP datasets, even when comparing gold standard results. There are several possible explanations for this behavior:

- **Insufficient data/subjects.** Some classes in the REALDISP data had only one training sample per subject for the algorithm to train on. Furthermore, the PD and REALDISP datasets had only 10 and 17 subjects respectively. Therefore, it is possible that these datasets are inadequate for modeling the various styles in which the different subjects perform certain tasks, particularly when one subject is removed for testing. That is, if there were little overlap between data from different subjects, data from training subjects would not be able to support data from the test subject.
- **Incorrect choice of points for high weighting.** STM may have chosen to give higher weight to training samples from undesired classes, resulting in decreased performance. It is possible that this behavior is due to class imbalance, incorrect parameters, or, in the case of REALDISP, an increased difficulty from the multiclass setting over the original binary scenario.
- **Variations in feature space.** The facial action unit detect tasks used much higher-dimensional features than the ones used on the REALDISP or PD datasets. It is possible that nature of the data in the higher-dimensional feature space (*i.e.*, distribution of classes and distribution of each subject's data) was particularly amenable to STM.

5.4 Analysis of multiclass STM on synthetic data

In order to explore these issues in more detail, we performed experiments on synthetic data, where we could control the various properties of the datasets. Recall that STM is a form of transfer learning. Therefore, it can only demonstrate improvement over baseline algorithms if the baseline assumption (that the training and test data come from the same distribution) is violated. Here, we focus on the linear version of STM because a personalized linear model can lead to the same performance as a kernel-based model, with the benefit of being



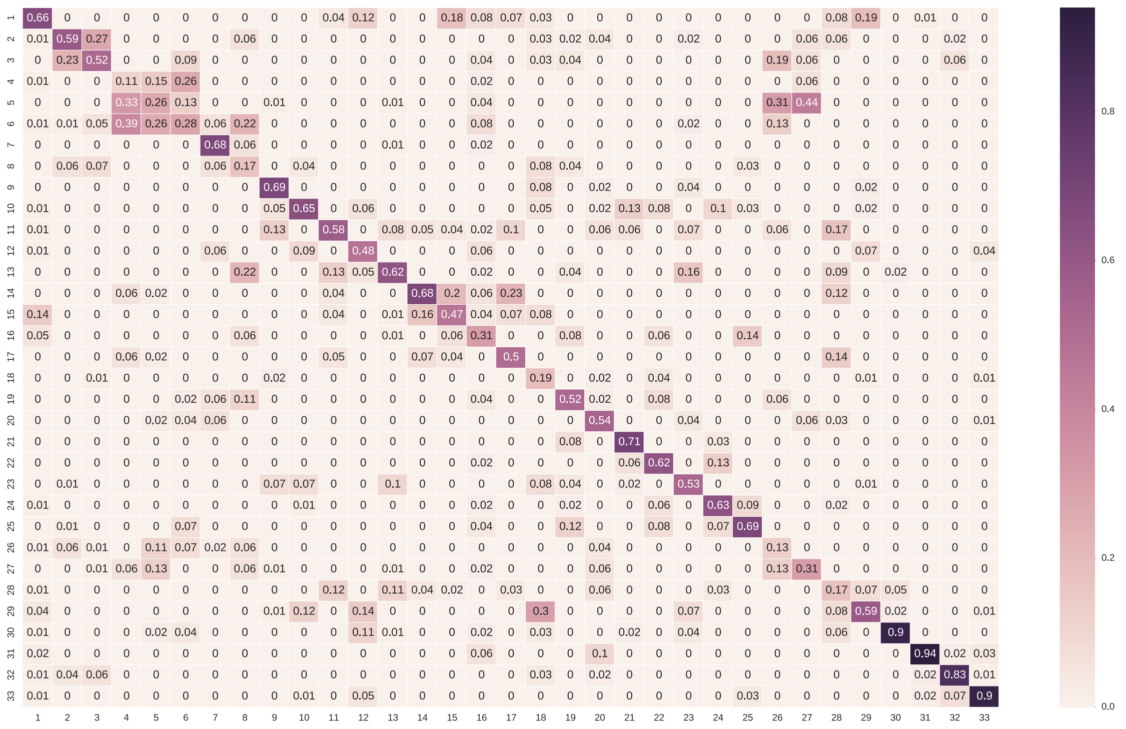


Figure 5.4: Confusion matrices for WW-SVM (top) and WW-STM (bottom). Columns are ground truth, rows are classifier output. Each column of the complete confusion matrices is normalized by the number of samples and sums to 1. Darker colors mean higher values. Perfect performance would show dark purple along the diagonal with off-white everywhere else.

Table 5.3: All possible combinations of the three conditions and the corresponding synthetic datasets

Conditions 1	2	3	**Corresponding datasets**
×	×	×/✓	**Not possible:** Failing condition #1 means the classes are easily separable in training data (combination of all subjects), which implies that data from each subject are easily separable, which implies that condition #2 *must* be met.
×	✓	✓	-
×	✓	×	**Dataset #1**
✓	×	✓	-
✓	×	×	**Dataset #2**
✓	✓	×	**Dataset #3**
✓	✓	✓	**Dataset #4a:** Equal distribution of classes. **Dataset #4b:** Unequal distribution of classes. **Dataset #4c:** Unequal distribution of classes, and data from test subjects are missing one (random) class.

much more light weight. In summary, we believe the following three conditions facilitate improvement of a linear STM over the baseline:

1. **Training data must be non-linearly separable.** This condition indicates that there is room for improvement from the baseline performance of a linear SVM.
2. **Test data from one subject *must* be linearly separable** That is, when using a linear SVM, training and testing within one subject must show a performance improvement over training and testing over the entire dataset. This improvement indicates that an optimal solution exists for a test subject with higher performance than the solution resulting from training and testing on the entire dataset.
3. **Data from different classes must have some overlap.** This condition differs slightly from Condition #1 above. If only Condition #1 holds, and there is no overlap between classes, then it is likely that KMM (the first iteration of STM) will reach the optimal solution. However, when classes overlap, it is possible for KMM to place high weight on points from the wrong class, and there is room for STM to improve performance.

There are eight possible combinations of meeting or failing the above three conditions, all listed in Table 5.3 (left). Of these eight possibilities, we created synthetic datasets from four, as summarized in Table 5.3. Data from 20 subjects were generated to create each of the following datasets. For Datasets #1-3 and 4a, 20 data points were generated for each of the three possible classes for all 20 subjects, leading to a total of 3000 data points in 2D space.

In all experiments that follow, the basic algorithm is a linear SVM. However, the principles can be applied to more complex forms of SVM, such as a version that uses an RBF kernel. Note that, for the REALDISP experiments reported in Section 5.3, STM was initialized to the SVM solution. However, on these synthetic datasets, initializing STM to the KMM solution produced better results.

5.4.1 Dataset #1

In this baseline dataset, the training data are easily separable by the algorithm (fails condition #1). Because the training data are a union of all the subjects (excluding the left out subject), each subject must also be easily separable (meets condition #2). In this scenario, the test data are drawn from the same distribution as the training data, which fulfills the assumptions of a standard SVM. Therefore, STM should not offer any improvement, regardless of whether the classes are overlapped or not.

The data from class k of subject i were generated from a 2D Gaussian distribution with mean vector μ_i^k and covariance matrix $\Sigma = 0.5\,I$ where I is the identity matrix. These centers were generated from Gaussian distributions themselves (one for each of the three classes):

$$\mu_i^1 \sim \mathcal{N}\left(\begin{bmatrix}-4\\-4\end{bmatrix}, \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right) \qquad \forall\,\text{subject } i$$

$$\mu_i^2 \sim \mathcal{N}\left(\begin{bmatrix}0\\4\end{bmatrix}, \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right) \qquad \forall\,\text{subject } i$$

$$\mu_i^3 \sim \mathcal{N}\left(\begin{bmatrix}4\\-4\end{bmatrix}, \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right) \qquad \forall\,\text{subject } i$$

where μ_i^1, μ_i^2, and μ_i^3, represent the cluster centers for classes 1, 2, and 3 respectively.

This synthetic dataset and results from SVM, KMM, and STM on a sample test subject are shown in Figure 5.5. Classification accuracies averaged across all synthetic test subjects were 0.98 for SVM, 1 for KMM, and 1 for STM. As expected, the three algorithms demonstrate similar performance.

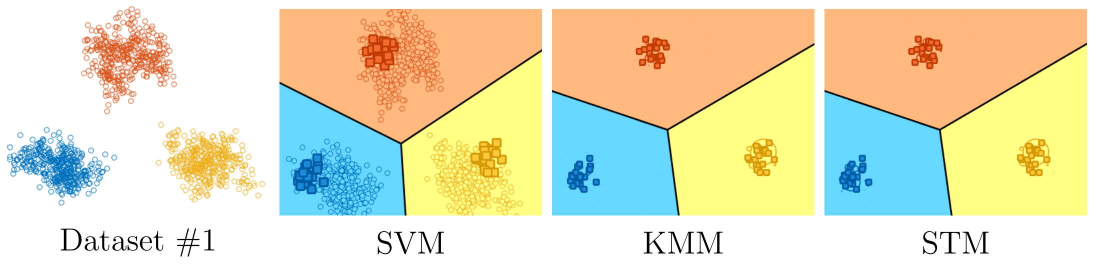


Figure 5.5: Results of SVM, KMM and STM on a sample test subject from Dataset #1. Each color represents one class. Training data are shown with circles. Test data are shown with squares. For KMM and STM, the size of the circles indicates the amount of weight placed on that training point.

5.4.2 Dataset #2

In this dataset, the training data are non-linearly separable (meets Condition #1), but the test data from each subject are also non-linearly separable (fails Condition #2). As with Dataset #1, the test data are drawn from the same distribution as the training data, fulfilling the assumptions of a standard SVM. Therefore, STM should not offer any improvement over SVM, regardless of whether the classes are overlapped or not. However, this dataset should show generally worse performance than Dataset #1.

For this dataset, 20 points from each class were generated for each subject. Each class was modeled using two Gaussian distributions, and each point was generated by randomly selecting one of the two Gaussians:

$$\begin{aligned}
\mathbf{x}^1 &\sim \mathcal{N}\left(\begin{bmatrix}-6\\-6\end{bmatrix}, \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right) + \mathcal{N}\left(\begin{bmatrix}-3\\0\end{bmatrix}, \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right)\\
\mathbf{x}^2 &\sim \mathcal{N}\left(\begin{bmatrix}0\\-6\end{bmatrix}, \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right) + \mathcal{N}\left(\begin{bmatrix}0\\6\end{bmatrix}, \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right)\\
\mathbf{x}^3 &\sim \mathcal{N}\left(\begin{bmatrix}-3\\0\end{bmatrix}, \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right) + \mathcal{N}\left(\begin{bmatrix}6\\-6\end{bmatrix}, \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right)
\end{aligned}$$

Here, $\mathbf{x}^1$, $\mathbf{x}^2$, and $\mathbf{x}^3$ represent points from classes 1, 2, and 3 respectively.

This synthetic dataset and results from SVM, KMM, and STM on a sample test subject are shown in Figure 5.6. Classification accuracies averaged across all synthetic test subjects were 0.50 for SVM, 0.53 for KMM, and 0.63 for STM. As expected, all three algorithms perform worse on this dataset than on Dataset #1. However, somewhat surprisingly, STM shows increased performance over SVM and KMM. We believe the this improvement is due to STM selectively choosing which test points to ignore so that it can fit to the majority. Indeed, in Figure 5.6, we can see that training points under the smaller yellow and blue test clusters have low weight.

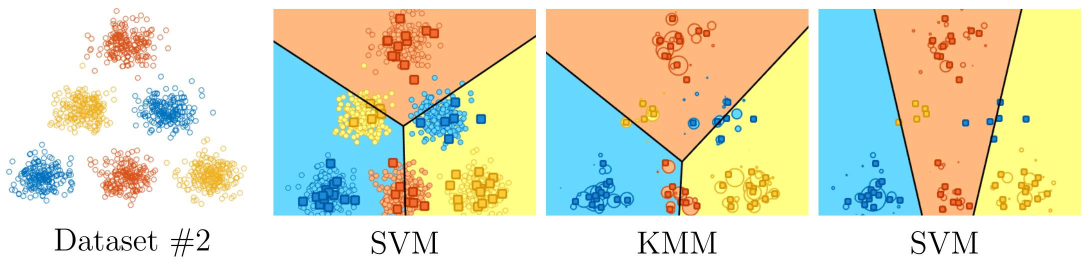


Figure 5.6: Results of SVM, KMM and STM on a sample test subject from Dataset #2. Each color represents one class. Training data are shown with circles. Test data are shown with squares. For KMM and STM, the size of the circles indicates the amount of weight placed on that training point.

5.4.3 Dataset #3

In this dataset, the training data are non-linearly separable (meets condition #1), but data from each individual test subject are linearly separable (meets condition #2). In this scenario, the test data are drawn from a *different* distribution than the training data, which fails the assumption of a standard SVM. Therefore, STM should offer some improvement. However, because the classes are not overlapped, the first iteration of STM, which is KMM, should reach the optimal or near optimal solution. Therefore, STM should not offer any improvement over KMM, although both STM and KMM should be better than SVM.

This dataset was generated similarly to Dataset #1. The data from class k of subject i were generated from a 2D Gaussian distribution with mean vector μ_i^k and covariance matrix $\Sigma = 0.5\,I$ where I is the identity matrix. These centers were generated from Gaussian distributions themselves (one for each of the three classes):

$$\begin{aligned}
\mu_i^1 &\sim \mathcal{N}\left(\begin{bmatrix}-6\\-6\end{bmatrix}, \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right) + \mathcal{N}\left(\begin{bmatrix}-3\\0\end{bmatrix}, \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right) & \forall\,\text{subject } i\\
\mu_i^2 &\sim \mathcal{N}\left(\begin{bmatrix}0\\-6\end{bmatrix}, \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right) + \mathcal{N}\left(\begin{bmatrix}0\\6\end{bmatrix}, \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right) & \forall\,\text{subject } i\\
\mu_i^3 &\sim \mathcal{N}\left(\begin{bmatrix}-3\\0\end{bmatrix}, \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right) + \mathcal{N}\left(\begin{bmatrix}6\\-6\end{bmatrix}, \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right) & \forall\,\text{subject } i
\end{aligned}$$

Readers may note the similarity to Dataset #2. Here, once the centers μ_i^k of each subject's Gaussians are chosen, *all* data from that subject for that class are generated according to a single Gaussian. For Dataset #2, however, data from one class of one subject come from two Gaussians.

This synthetic dataset and results from SVM, KMM, and STM on a sample test subject are shown in Figure 5.7. Classification accuracies averaged across all synthetic test subjects were 0.53 for SVM, 0.98 for KMM, and 1 for STM. As expected, STM and KMM have similar performance and show improvement over SVM.

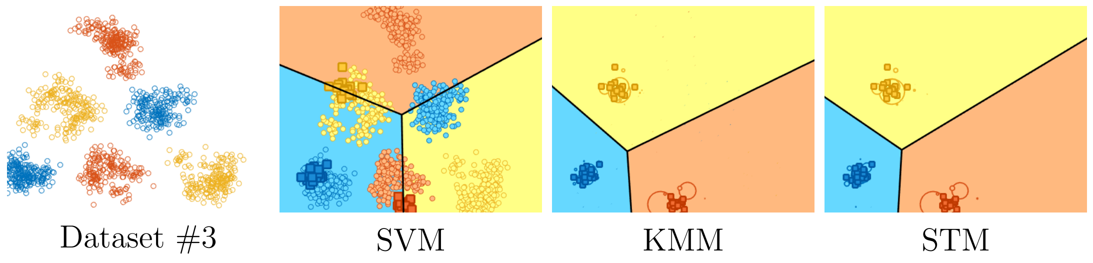


Figure 5.7: Results of SVM, KMM and STM on a sample test subject from Dataset #3. Each color represents one class. Training data are shown with circles. Test data are shown with squares. For KMM and STM, the size of the circles indicates the amount of weight placed on that training point.

5.4.4 Dataset #4

This section describes three versions of Dataset #4. All fulfill the three conditions described at the beginning of Section 5.4, but they are designed to represent increasing realism and difficulty.

Version A: equal distribution of classes

This dataset is similar to Dataset #3, except the classes are overlapped. The overlapping causes KMM to occasionally place higher weight on points from the incorrect class. Here, STM should offer improvement because it can, over several iterations, selectively choose points from the correct class.

This dataset was generated in the same way as Dataset #3 except the Gaussians used to generate each subject's data had larger covariance ($\Sigma = I$) and Gaussian centers for generating each subject's mean vector μ_i^k were moved closer to each other:

$$\mu_i^1 \sim \mathcal{N}\left(\begin{bmatrix}-4\\-4\end{bmatrix}, \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right) + \mathcal{N}\left(\begin{bmatrix}-2\\0\end{bmatrix}, \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right) \qquad \forall \text{subject } i$$
$$\mu_i^2 \sim \mathcal{N}\left(\begin{bmatrix}0\\-4\end{bmatrix}, \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right) + \mathcal{N}\left(\begin{bmatrix}0\\4\end{bmatrix}, \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right) \qquad \forall \text{subject } i$$
$$\mu_i^3 \sim \mathcal{N}\left(\begin{bmatrix}-2\\0\end{bmatrix}, \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right) + \mathcal{N}\left(\begin{bmatrix}4\\-4\end{bmatrix}, \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right) \qquad \forall \text{subject } i$$

These modifications ensured higher overlap between classes.

This synthetic dataset and results from SVM, KMM, and STM on a sample test subject are shown in Figure 5.8a. Classification accuracies averaged across all synthetic test subjects were 0.54 for SVM, 0.78 for KMM, and 0.80 for STM. Surprisingly, results do not show a large performance improvement of STM over KMM. It was found that, on some of the synthetic subjects, STM was unable to sufficiently move the the decision boundary and converged to a solution that divided a cluster (see Figure 5.9a). On other synthetic subjects, STM would initialize with higher weighted points distributed among two classes instead of three. Over iterations, the points in this third class would be assigned progressively smaller weight until that class was no longer considered (see Figure 5.10a).

Version B: unequal distribution of classes

This dataset uses the same distributions as Dataset #4a to generate each subject's data. While all datasets described above generated an equal number (20) of samples per subject, per class, this dataset was built using a different number of samples for each class. In this way, this synthetic dataset can better represent real data, where different classes often occur with different frequency. The number of samples generated was as follows:

- Blue class – 35 samples/subject
- Red class – 20 samples/subject
- Yellow class – 5 samples/subject

Results are shown in Figure 5.8b. Classification accuracies averaged across all synthetic test subjects were 0.58 for SVM, 0.87 for KMM, and 0.88 for STM. Similar to Dataset #4a, STM demonstrates little improvement over KMM.

Version C: missing classes

This dataset was generated in the same manner as Dataset #4b. However, during leave-one-subject-out cross validation, all samples from one randomly chosen class of the test subject's data were removed. This dataset represents the scenario where test subjects are missing classes. Several subjects in REALDISP are missing classes (see Figure 5.2). This situation can also occur in real-life scenarios. For example, suppose a user wants to a wearable device to track running and walking, but never swims. And suppose the user's device tracks running, walking, swimming and cycling. Such a user would therefore be missing samples from the swimming class.

Results are shown in Figure 5.8c. Classification accuracies averaged across all synthetic test subjects were 0.53 for SVM, 0.87 for KMM, and 0.89 for STM. Again, STM demonstrates only modest improvement over KMM, displaying the same behavior as with Dataset #4a (either failing to move the decision boundary or removing a class from consideration).

5.5 Improving the STM objective function constraints

As discussed above, poor performance by STM is often due to an incorrect choice of weighting. Figures 5.9a and 5.10a show two examples of incorrect weighting leading to poor solutions on Dataset #4a. It is possible that better methods of importance reweighting would improve performance. This section describes two novel adjustments to the STM objective function constraints. These modifications were designed to better describe the desired solution for which points should be assigned higher weight.

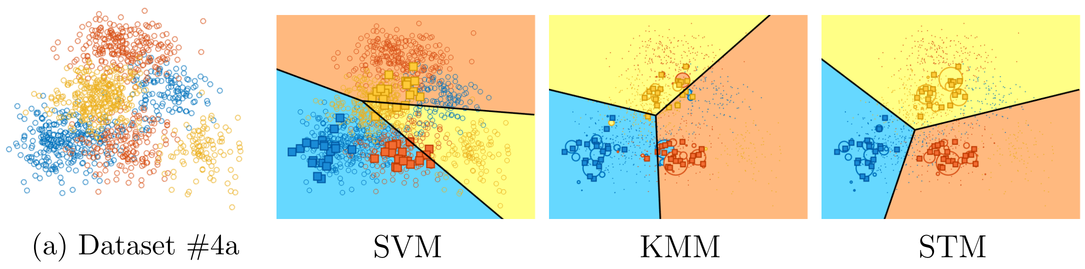
5.5.1 Equal weighting of classes

One solution to the issue of STM removing a class from consideration is to put constraints on the KMM portion of STM such that each class must have (approximately) equal total weight. Recall from Section 4.1.3 that solving for $\mathbf{s}$ involves solving the following quadratic programming problem:

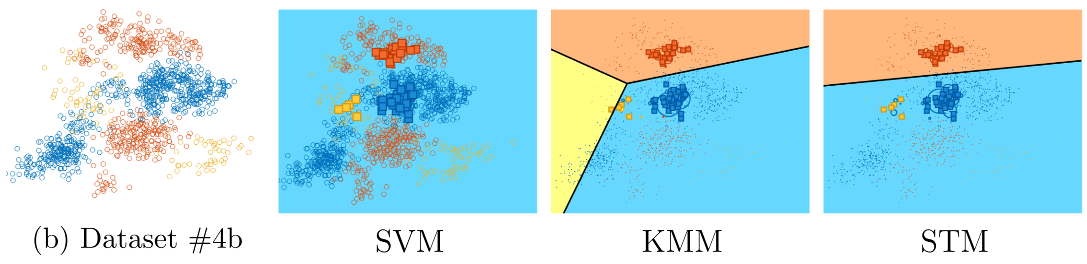
$$\begin{aligned} & \min_{\mathbf{s}} \mathbf{s}^\top K \mathbf{s} + (\boldsymbol{\ell} - \boldsymbol{\kappa})^\top \mathbf{s} + \text{const.} \\ \text{subject to} \quad & 0 \le s_i \le B, \qquad \forall i \\ & \left| \frac{1}{n} \sum_{i=1}^{n} s_i - 1 \right| \le \varepsilon \end{aligned}$$

The second condition enforces the average value of s_i to be $1 \pm \varepsilon$.

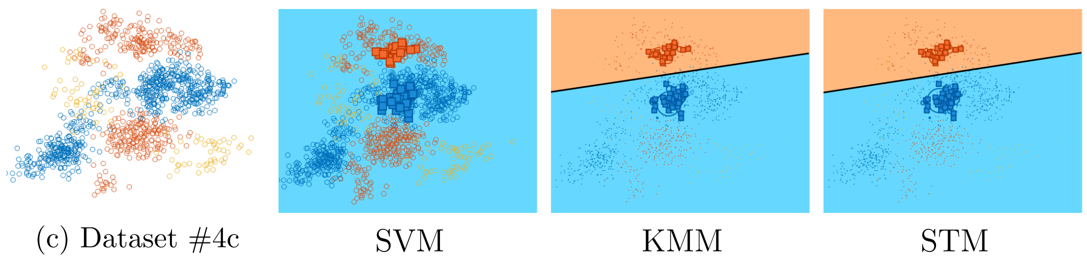
Let n_{tr} be the total number of training points, m be the number of classes, n_j be then number of points in the jth class, and s_i^j be the weight of the ith point in the jth class. In



(a) Dataset #4a



(b) Dataset #4b



(c) Dataset #4c

Figure 5.8: Results of SVM, KMM and STM on a sample test subject from Datasets #4a-c. Each color represents one class. Training data are shown with circles. Test data are shown with squares. For KMM and STM, the size of the circles indicates the amount of weight placed on that training point.

order to equalize the total weight of each class, we want the sum of all weights from one class to be n_{tr}/m. Therefore, the average weight on points in each class should be within ε of $(n_{\text{tr}}/m)/n_j$, and our new constraint is as follows:

$$\left| \frac{1}{n_j} \sum_{i=1}^{n_j} s_i^j - \frac{n_{\text{tr}}/m}{n_j} \right| \leq \varepsilon, \qquad \forall j.$$

This inequality simplifies to

$$\begin{aligned} \sum s_i^j &\leq \frac{n_{\text{tr}}}{m} + n_j \varepsilon, \qquad \forall j \\ -\sum s_i^j &\leq -\frac{n_{\text{tr}}}{m} + n_j \varepsilon, \qquad \forall j. \end{aligned}$$

5.5.2 Proportional weighting of classes

Another solution is to put constraints on the KMM portion of STM such that the *proportion* of weight assigned to each class is the same as the proportion of samples belonging to that class. This constraint is equivalent to restricting the average weight within *each class* to be within ε of 1. That is,

$$\left| \frac{1}{n_j} \sum_{i=1}^{n_j} s_i^j - 1 \right| \leq \varepsilon, \qquad \forall j.$$

This inequality simplifies to

$$\begin{aligned} \sum s_i^j &\leq n_j(1+\varepsilon), \qquad \forall j, \\ -\sum s_i^j &\leq -n_j(1+\varepsilon), \qquad \forall j. \end{aligned}$$

5.5.3 Results

Figures 5.9b and 5.10b demonstrate the improved performance of these modifications on two sample subjects of Dataset #4a. As expected, because Dataset #4a exhibits balanced classes, the two modifications are equivalent. On average, the improved STMs achieved an accuracy of 0.92 on Dataset #4, compared to the 0.80 of the original STM.

Table 5.4 compares the performance of all algorithms on Dataset #4b. Figure 5.11 shows the separating hyperplanes learned by the original, equal weighting, and proportional weighting STM algorithms on subject #2. We can see that the original STM chooses to assign more weight to training samples of the *blue* class that are near the test subject's yellow samples. The equal weighting method successfully chooses to give more weight to yellow training samples near the same cluster, but assigns too much weight to these yellow points, pushing the margin between the yellow and blue clusters too far. The proportional weighting method, however, assigns appropriate weights to the training data, learning a hyperplane that only misclassifies one point. On average, the proportional weighting STM demonstrates a large performance improvement on this dataset, with an accuracy of 0.96 compared to 0.88 and 0.89 for the original and equal weighting STMs respectively.

Table 5.5 compares the performance of all algorithms on Dataset #4c. Figure 5.12 shows the separating hyperplanes learned by the three STM algorithms on subject #2. Unfortunately, neither of the modified STM algorithms are able to demonstrate a large improvement over KMM. Both modifications make assumptions about the class distribution of the test subject's data: the equal weighting method assumes the test subject's samples are evenly distributed between all classes, and the proportional weighting method assumes the test subject's samples have the same class distribution as the training data. In Datasets #4a and b, these assumptions are valid, and the algorithms perform well. However, in Dataset #4c, which represents many realistic scenarios, these assumptions are violated. Results on all versions of Dataset #4 indicate that in situations where the distribution of the test subject's classes can be estimated, these modified STM algorithms have the potential to demonstrate improved performance over KMM and SVM. However, they do not perform better when no assumptions can be made about the test subject's data.

Table 5.6 compares the accuracy of SVM, KMM, the original STM, and the proportional weighting STM on the REALDISP dataset. The equal weighting method was not included because it was shown to not perform well on datasets where the class distribution was unbalanced, as is the case for REALDISP (see Figure 5.3 left). The parameter values of C and λ were fixed to be those that most frequently led to the gold-standard results shown in Table 5.2 across all subjects. The overall improvement of the proportional weighting method over the original STM demonstrates that this idea helps STM to select the correct points for assigning higher weight. Unfortunately, on this dataset, the performance was not substantially better than a generic SVM.

5.6 Analyzing the STM objective function

The lack of improvement from all personalization methods (KMM, and two versions of STM) over a generic SVM, as shown in Table 5.6, was surprising. I hypothesized that the REALDISP dataset did not satisfy condition #2 described in Section 5.4: that the data from a single subject be linearly separable. Therefore, an SVM was trained and tested on each individual subject to determine whether a set of hyperplanes could separate the 33 classes in the 6-dimensional feature space. Results are shown under "ideal" in Table 5.6. The high accuracy (0.9 on average) indicates that linear hyperplanes are indeed able to separate the classes in this feature space. It was therefore perplexing that the KMM and STM algorithms were not able to get closer to this ideal solution.

I then hypothesized that the STM objective functions may not have a local optimum at this "ideal" solution. *I.e.*, there may not exist a set of weights on the training data that would support that set of hyperplanes. To test this hypothesis, both STM algorithms were initialized to the ideal solution. If the performance from the ideal initialization of

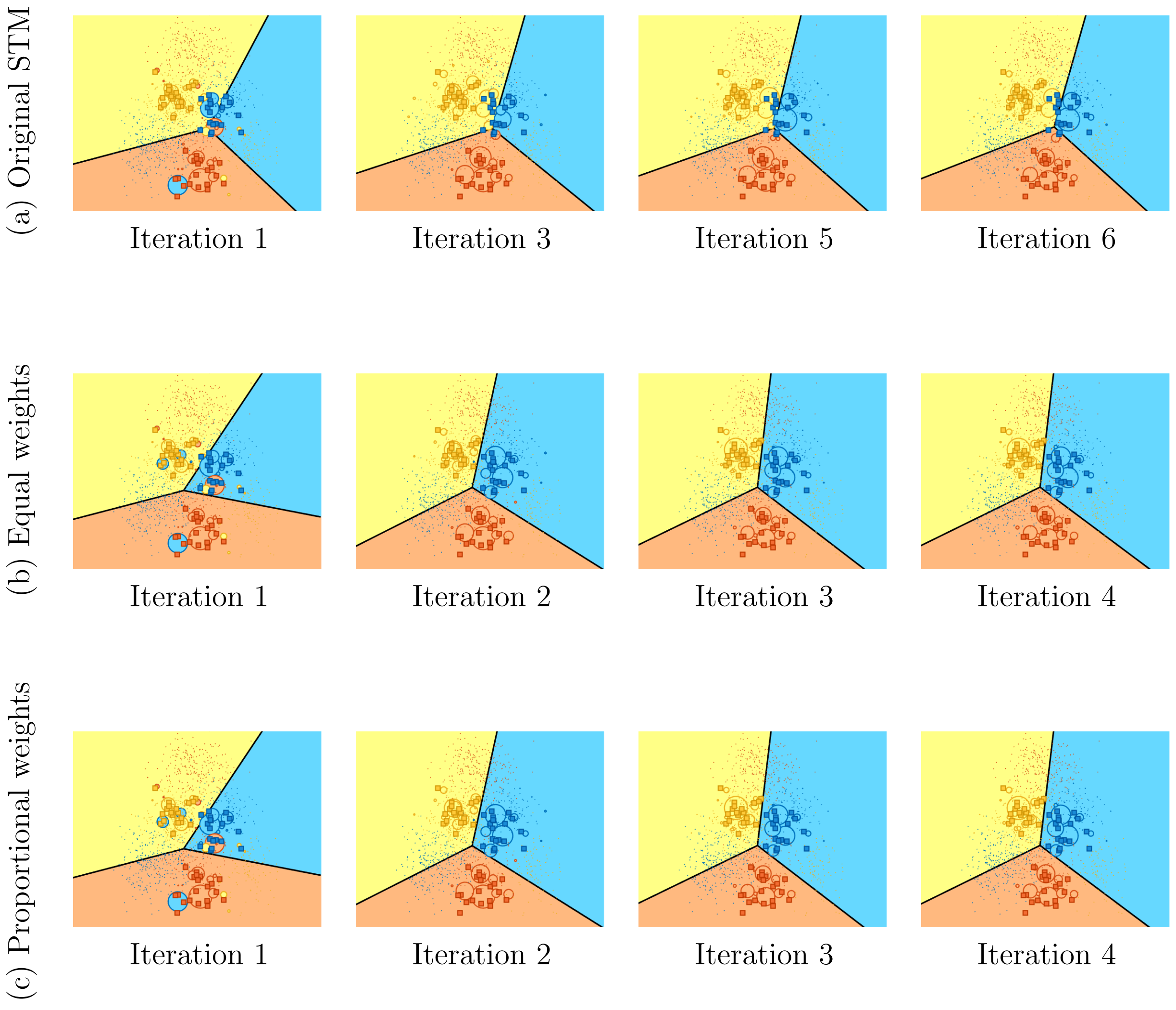


Figure 5.9: Comparison of (a) original and (b-c) improved versions of STM on data from Subject #18 of Dataset #4a. As expected, the equal weighting and proportional weighting variants are equivalent when the class distribution is balanced. Classification accuracies after convergence are 83.3% and 100% from the original and improved STM algorithms, respectively.

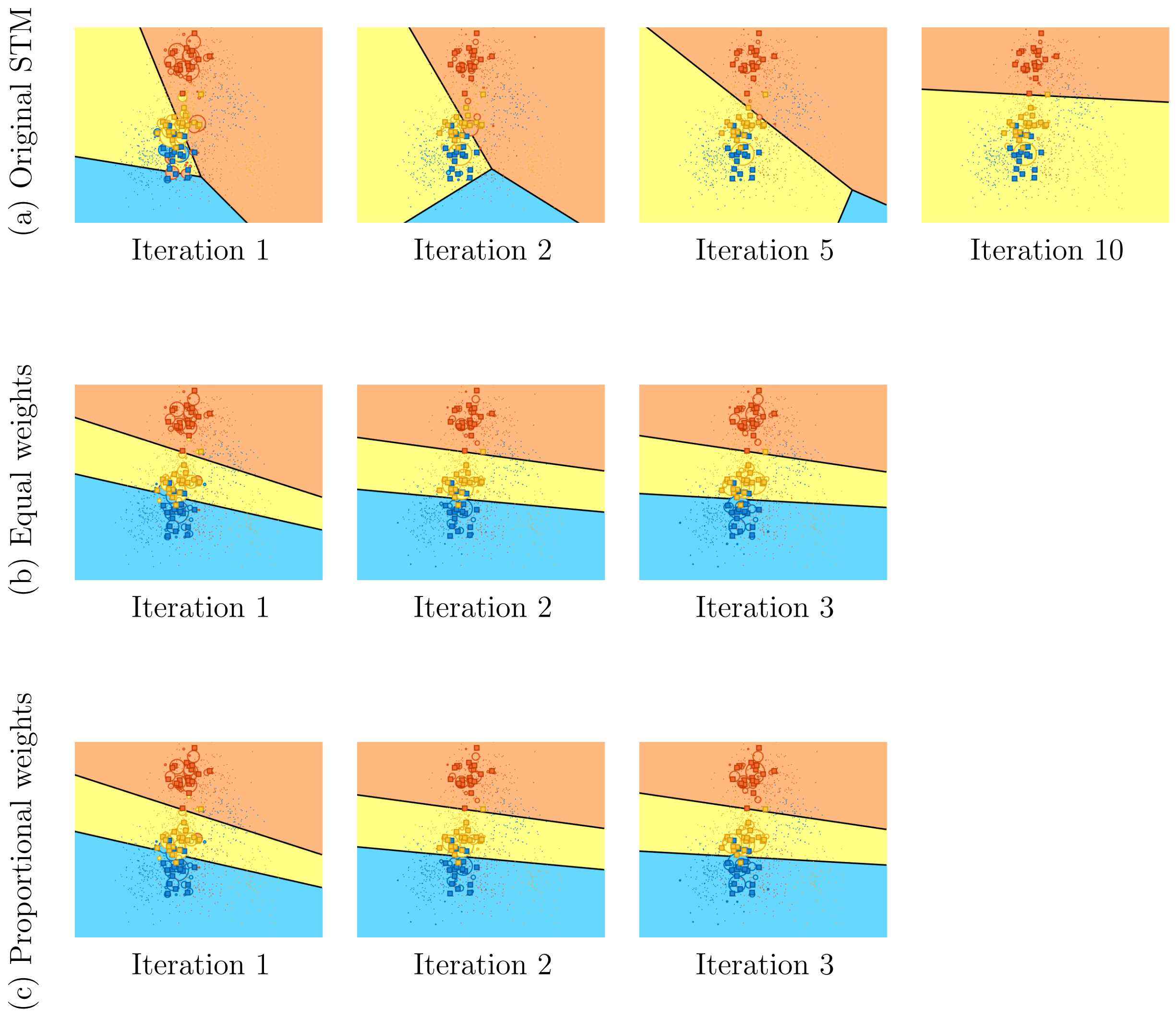


Figure 5.10: Comparison of (a) original and (b-c)) improved versions of STM on data from Subject #8 of Dataset #4a. As expected, the equal weighting and proportional weighting variants are equivalent when the class distribution is balanced. For this subject, both improved STM algorithms converged in three iterations. Classification accuracies after convergence are 65.0% and 90.0% from the original and improved STM algorithms respectively.

Table 5.4: Accuracy of SVM, KMM and three versions of STM on Dataset #4b

			STM		
Subject	**SVM**	**KMM**	**Original**	**Equal weights**	**Proportional weights**
1	0.58	0.97	0.97	0.93	0.98
2	0.58	0.93	0.92	0.93	0.98
3	0.58	1.00	1.00	1.00	1.00
4	0.58	1.00	1.00	1.00	1.00
5	0.58	1.00	1.00	1.00	1.00
6	0.58	0.92	0.92	1.00	0.97
7	0.58	0.92	0.92	0.95	1.00
8	0.58	0.97	1.00	1.00	1.00
9	0.58	1.00	1.00	0.88	1.00
10	0.58	1.00	1.00	1.00	1.00
11	0.58	0.87	0.92	0.92	0.92
12	0.58	0.90	0.88	0.97	0.88
13	0.58	0.75	0.75	0.55	0.85
14	0.58	0.92	0.92	1.00	0.92
15	0.58	0.42	0.42	0.75	0.92
16	0.58	0.32	0.33	0.57	0.90
17	0.58	1.00	1.00	0.80	1.00
18	0.58	0.55	0.58	0.58	0.92
19	0.58	1.00	1.00	1.00	1.00
20	0.58	0.97	1.00	0.98	1.00
Mean	**0.58**	**0.87**	**0.88**	**0.89**	**0.96**

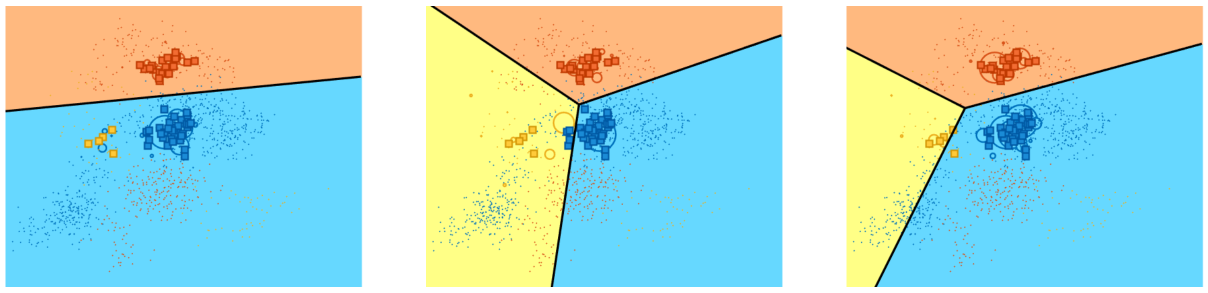
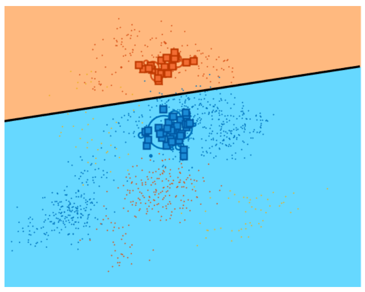
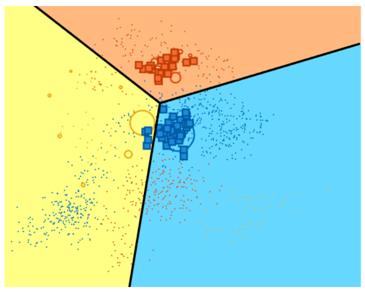

(a) Original (b) Equal weights (c) Proportional weights

Figure 5.11: Solutions for three versions of STM on Dataset #4b, Subject 2

Table 5.5: Accuracy of SVM, KMM and three versions of STM on Dataset #4c

			STM		
Subject	**SVM**	**KMM**	**Original**	**Equal weights**	**Proportional weights**
1	0.64	0.96	1.00	0.76	1.00
2	0.88	0.93	0.88	0.82	0.93
3	0.64	1.00	1.00	0.85	1.00
4	0.88	1.00	1.00	1.00	1.00
5	0.00	1.00	1.00	1.00	1.00
6	0.00	0.80	0.80	0.80	0.68
7	0.88	0.88	0.88	0.82	0.97
8	0.64	1.00	1.00	0.95	1.00
9	0.88	1.00	1.00	0.80	1.00
10	0.88	1.00	1.00	0.90	1.00
11	0.00	0.80	0.80	0.92	0.76
12	0.64	0.98	0.96	0.96	0.98
13	0.00	0.80	0.80	0.48	0.64
14	0.88	0.88	0.88	0.88	0.88
15	0.64	0.36	0.36	0.67	1.00
16	0.64	0.38	0.36	0.73	1.00
17	0.64	1.00	1.00	0.96	1.00
18	0.88	1.00	1.00	0.95	0.97
19	0.00	1.00	1.00	0.84	0.56
20	0.00	0.68	1.00	0.60	0.44
Mean	**0.53**	**0.87**	**0.89**	**0.84**	**0.89**


(a) Original


(b) Equal weights

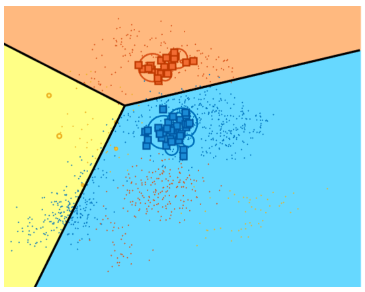

(c) Proportional weights

Figure 5.12: Solutions for three versions of STM on Dataset #4c, Subject 2

Table 5.6: Accuracy of SVM, KMM, Original STM, and Proportional STM on REALDISP

				Original STM		**Proportional STM**	
Subject	**SVM**	**Ideal**	**KMM**	**Original init.**	**Ideal init.**	**Original init.**	**Ideal init.**
1	0.42	0.94	0.39	0.37	0.94	0.47	0.75
2	0.42	0.93	0.29	0.29	0.93	0.50	0.68
3	0.47	0.76	0.45	0.45	0.76	0.56	0.61
4	0.13	0.89	0.25	0.25	0.87	0.32	0.57
5	0.36	0.97	0.35	0.34	0.97	0.45	0.81
6	0.38	0.87	0.38	0.38	0.87	0.43	0.72
7	0.36	0.82	0.31	0.30	0.71	0.27	0.47
8	0.31	0.91	0.24	0.24	0.92	0.09	0.76
9	0.33	0.74	0.22	0.20	0.74	0.37	0.59
10	0.54	0.92	0.32	0.38	0.93	0.54	0.75
11	0.19	0.90	0.17	0.17	0.92	0.25	0.57
12	0.50	0.93	0.41	0.42	0.93	0.55	0.71
13	0.33	0.90	0.29	0.29	0.90	0.39	0.76
14	0.20	0.98	0.31	0.31	0.98	0.38	0.65
15	0.32	1.00	0.05	0.05	1.00	0.00	0.42
16	0.56	0.92	0.47	0.47	0.89	0.51	0.67
17	0.46	0.94	0.49	0.51	0.94	0.51	0.73
Mean	**0.37**	**0.90**	**0.32**	**0.32**	**0.89**	**0.39**	**0.66**

STM was similar to the ideal performance, it would indicate that a local optimum existed near that solution. Results are shown in Table 5.4 under “Ideal init.” We can see that the original STM maintains very similar performance to the ideal solution (0.89 compared to 0.90), which indicates that it was able to find a set of weightings to support the ideal hyperplanes. The proportionally weighted STM, however, demonstrates a significant drop in performance (down to 0.66), which indicates that the constraints on the sample weights were too restrictive to support the ideal hyperplanes.

5.6.1 Experiments with global optimization

Given that the ideal solution was a local optimum (at least for the original STM), another hypothesis for the lack of improvement was that the algorithm was getting stuck in other local optima. Convex functions are guaranteed to have a single optimum, but biconvex functions like STM may have many stationary points, including saddle points, in addition to local optima [44]. The solution, therefore, was to explore global optimization methods of biconvex functions.

BARON is a solver for global optimization of mixed-integer nonlinear optimization prob-

Table 5.7: Comparing solutions from ideal versus regular initialization to global optimum solution from BARON

Optimizer	Accuracy	Objective Function Value	Objective function component values		
			$\frac{1}{2}\sum_j \|\mathbf{w}_j\|^2$	$\frac{1}{2}\sum_i \mathbf{s}_i L(\mathbf{x}_i, y_i)$	$\frac{1}{{n_{\mathrm{tr}}}^2}\mathbf{s}\top\mathbf{K}\mathbf{s} - \frac{2}{{n_{\mathrm{tr}}}^2}\boldsymbol{\kappa}\top\mathbf{s}$
Ideal init.	0.78	1.60	3.35	0.0580e-4	0.5556
Original init.	0.68	2.06	14.08	0.1141e-4	1.1253
BARON	0.33	0.27	0.61	0.0000e-4	0.0636

lems [73]. Unfortunately, all values of the $\mathbf{W}$ matrix and all weights s_i (one per training sample) are considered as variables, making the STM problem on the REALDISP dataset too complex for BARON. Therefore, Dataset #4a was used instead to test whether the global optimum of the STM objective function corresponded to high accuracy. The accuracy, objective function total value, and objective function component values are compared in Table 5.7 for the ideal and regular initializations of STM, as well as a solution found by BARON. Note that BARON terminated due to time out and did not actually find the global optimum. Nonetheless, the solution found by BARON has an accuracy of only 0.33 (suggesting that the solution classifies everything as a single class), while having the lowest objective function value. Furthermore, every *component* of the objective function has a lower value than those found in the ideal initialization solution, which has the highest accuracy. Even though BARON did not find the global optimum, these results indicate that minimizing the STM objective function does not monotonically increase the accuracy.

Global optimization of STM could be explored further. It is possible that, even though the objective function is not monotonically related to accuracy, the solution at the global optimum has the highest accuracy. Building an even smaller toy example and finding the global optimum could shed light on this question. Further experiments on how the dataset affects the shape of the objective function could inform which datasets would most benefit from STM over generic algorithms.

5.7 Conclusions

This chapter presents an in-depth comparison of SVM, KMM, and STM on the REALDISP dataset and several synthetic datasets. Previous work on STM reported performance improvements compared to SVM and KMM [25] on several facial action unit detection datasets. While it is possible that the feature spaces used in this chapter were not amenable to STM, the relative performance of these algorithms on the datasets in this chapter is consistent with what was found on the PD dataset, described in Chapter 4.3. Analysis into the STM solutions on synthetic data indicated that the algorithm failed to assign higher weight to training samples of the correct class. Therefore, two extensions to STM were developed to better describe which samples should be given higher weight. These extensions were found to have high performance on the synthetic datasets, but not on REALDISP. Further analysis into the STM objective function found that, while STM has a stable point (possibly a local optimum) near the ideal separating hyperplane of each subject, minimizing the objective function does not generally improve accuracy. These findings suggest that the STM objective function is not appropriate for these datasets using these feature representations. Other feature sets or personalization algorithms, such as those presented in Chapter 2.2, could lead to greater performance improvements over the generic SVM.

Chapter 6

Stratified Weakly Supervised Learning

Previous chapters explored the challenges of building a model for PD symptom detection that can generalize to (*i.e.*, maintain high performance on) a subject whose data were not in the training set. This chapter focuses on another challenge of PD symptom detection: maintaining high performance "*in the wild*." Ideally, machine learning models for continuous monitoring of PD motor symptoms "in-the-wild" would be trained on data collected in-the-wild so that the training data would most accurately mimic the test data. However, collecting accurate labels in-the-wild is prohibitively time consuming and labor intensive: either participants would need to be videotaped extensively so that a researcher could review the data and label it, or each participant would need to self-report on the exact starts and ends of each of their symptoms. Therefore, the most popular strategy for collecting labeled training data is to have participants perform scripted activities in controlled laboratory settings where video data can be used to label symptom occurrences (see Table 2.4). Researchers assume that these models trained on laboratory data will generalize to activities in the wild.

Although accurate labels in the wild are unrealistic, it is feasible for participants to record the approximate amount of tremor they experience within five-minute segments of time dispersed throughout the day. These approximate labels are called *weak labels*, and they inherently provide less information than the traditional accurate labels. While weakly supervised learning methods are relatively well-explored (see [5] for a review), few have applied these methods to PD symptom detection. In [40], Fisher et al. used a neural network to detect dyskinesia in weakly labeled data collected in patient homes. However, the authors applied the labels of their one-hour time intervals to every sub-interval. That is, if a patient reported dyskinesia during a one-hour segment, the authors assumed that dyskinesia occurred during the *entire* segment. This naive approach introduces many false positive labels. In this chapter, I compare the naive approach to several algorithms that explicitly account for the fact that only a portion of a positive segment is the event of interest.

Das et al. [32] compared several weakly supervised learning techniques on in-home data collected from two patients. One patient suffered from dyskinesia and the other had tremor.

Four days of data were collected with labels provided by patient diaries. While this work was a great proof-of-concept for the utility and necessity of weakly-supervised learning for in-home data, the number of subjects was limited (only one per symptom type) and there was no way to verify that symptom detections from trained algorithms were true symptom occurrences because there were no ground truth labels.

In this chapter, I explore and evaluate the usage of various weakly supervised learning algorithms for PD tremor detection. In particular, I assess the effect of weak labels on algorithm performance. Using accurate labels from the laboratory data (see Chapter 3.1), different levels of weak supervision are *simulated* by controlling the length of time segments for which labels are provided. In this way, the change in algorithm performance as label uncertainty increases can be analyzed. Furthermore, because ground truth labels exist, I can confirm whether algorithms trained on weakly labeled data are able to discriminate tremor from non-tremor.

Additionally, I introduce a new method for weakly supervised learning, which I call "*stratified weakly supervised learning.*" Traditional approaches use labels that indicate whether a single positive sample (instance of a PD symptom) exists within a segment. The algorithms then only look for a single instance within each positively labeled segment. This assumption may hold true for some applications, such as labeling images as containing or not containing a particular object of interest. However, it can be highly inaccurate for PD symptom detection, where a participant can experience symptoms for none of the segment, all of the segment, or any proportion in between. In this chapter, I show that symptom detection accuracy can be improved by using more nuanced, "stratified" labels and modifying several weak learning algorithms to make use of such labels. Some of the findings in this chapter were published in the proceedings of the 2017 Engineering in Medicine and Biology Conference [138].

6.1 Algorithms

This chapter compares the performance of several different weakly supervised learning algorithms. In particular, the top three performing algorithms reported by Das et al. [32] – Multiple Instance Support Vector Machine (MI-SVM) [6], Iterative Discriminative Axis Parallel Rectangle (ID-APR) [34] and Expectation Maximization Diverse Density (EM-DD) [139] – are compared to a naive Support Vector Machine (Naive-SVM), which applies segment-level labels to every subsegment within. *I.e.*, if a segment is labeled as containing some tremor, Naive-SVM assumes the entire segment consists of tremor, which is the strategy chosen Fisher et al. [40].

This chapter also presents a modification to the MI-SVM and ID-APR algorithms that allows them to take advantage of knowing the approximate amount of tremor within a segment given by stratified segment labels. In the LAB data, the incidence of tremor varies greatly across subjects (from 1.2% to 80.0% occurrence), as shown in Table 3.2 in Chapter 3.1. The MI-SVM and ID-APR algorithms are designed to find rare positive examples among many negative examples. As Table 3.2 shows, this assumption holds well for subject 12 (right hand), where only 1.2% of the data collected was tremor, but fails for subject 2 (left hand), where 80% was tremor. For subjects with frequent tremor, these algorithms may have low recall because they will classify very few events as tremor.

Before describing these algorithms, I first define some notation. Let $\{\mathbf{X}_i, y_i\}_{i=1}^{N_{\text{seg}}}$ be the set of time segments and their associated labels. Let $\{\mathbf{x}_i^j\}_{j=1}^{n_i}$ be the set of all events in $\mathbf{X}_i$. For the following five standard algorithms, $y_i \in +1, -1$, and a segment $\mathbf{X}_i$ is labeled as positive if there exists an event $\mathbf{x}_i^j \in \mathbf{X}_i$ that is positive.

6.1.1 Naive Support Vector Machine (Naive-SVM)

Given time segment labels, this algorithm makes the naive assumption that the segment-level label applies to every event within (see Figure 6.3b for an example). That is, it applies the label y_i to all $\mathbf{x}_i^j \in \mathbf{X}_i$. The objective function is therefore only a slight modification from Eq. 4.1:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{2}\sum_{i=1}^{N_{\text{seg}}}\sum_{j=1}^{n_i} L(y_i, f(\mathbf{x}_i^j)). \tag{6.1}$$

As before, I use the squared hinge-loss function: $L(y_i, f(\mathbf{x}_i)) = \max\{1 - y_i \cdot f(x_i), 0\}^2$, and this algorithm is implemented using the LIBLINEAR SVM library [37].

Naive-SVM is meant to serve as a baseline against the advantage of using more sophisticated, weakly supervised algorithms can be measured.

6.1.2 Multiple Instance SVM (MI-SVM)

This algorithm was originally developed by Andrews et al. [6]. It is similar to a standard SVM except that it does not make the assumption that all events within a segment share the label of the segment. Instead, it tries to find a separating margin such that at least one positive event in each positive segment is classified as positive, and all events in negative segments are classified as negative. Formally, MI-SVM minimizes the following objective function:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{2}\sum_{i}\left(\delta[y_i = +1]\,\xi_i^+ + \delta[y_i = -1]\sum_{j}\xi_{i,j}^-\right)$$

subject to

$$\begin{array}{ll}
\max_{\mathbf{x}_i^j \in \mathbf{X}_i}\left(\mathbf{w}^\top\mathbf{x}_i^j + b\right) \geq +1 - \xi_i^+, & \forall\, y_i = +1,\\
\xi_i^+ \geq 0, & \forall\, i,\\
\mathbf{w}^\top\mathbf{x}_i^j + b \leq -1 + \xi_{i,j}^-, & \forall \mathbf{x}_i^j \in \mathbf{X}_i,\ \forall\, y_i = -1,\\
\xi_{i,j}^- \geq 0, & \forall\, i, j.
\end{array}$$

This objective function is optimized iteratively. An SVM is first trained on all events in the negative segments and the mean of each positive segment. Given this initial separating hyperplane, positive selector variables are chosen to be the event with the maximum score in each segment. All events in negative segments are chosen to be negative selector variables because negatively labeled segments have no positive events by definition. A new SVM is trained on the selector variables, and this process is repeated until either the set of selector variables no longer changes or the maximum number of iterations has been reached. See Figure 6.3c for an example of how MI-SVM chooses selector variables. Note that this algorithm has no guarantee to converge and occasionally gets stuck in a loop of two (or more) sets of selector variables. I refer readers to [6] for more details on how to optimize this objective function. My implementation of MI-SVM was built around LIBLINEAR SVM from [37].

6.1.3 Iterative Discriminative Axis Parallel Rectangle (ID-APR)

This algorithm was originally developed by Dietterich et al. [34] and is the seminal work on multiple instance learning. The basic idea is to find the smallest axis parallel rectangle (APR), which constitutes a set of lower and upper bounds (one for each feature dimension), such that for each positive segment, there exists at least one event that lies within the APR.

The smallest APR is found over iterations. The APR is first initialized to the minimax APR: For each feature dimension, let a be the maximum of the minima from each segment and let b be the minimum of the maxima from each segment:

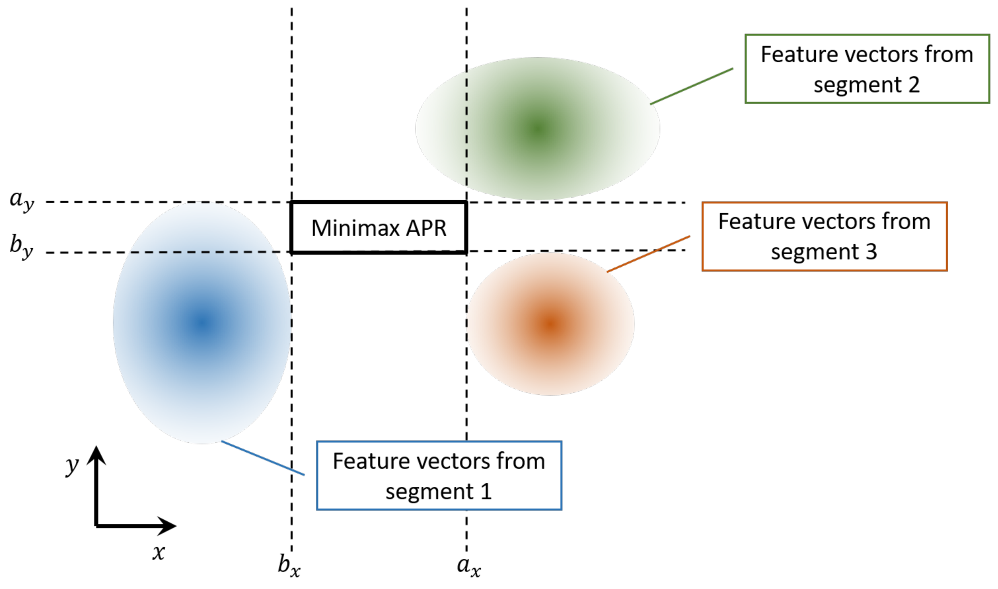
$$\begin{aligned} a &= \max_i \min_j \mathbf{x}_i^j \in \mathbf{X}_i \\ b &= \min_i \max_j \mathbf{x}_i^j \in \mathbf{X}_i. \end{aligned}$$

Then, the lower bound of the minimax APR is $\min(a, b)$ and the upper bound is $\max(a, b)$. Note that it is possible for the minimax APR to not contain a single point from one of the positive segments, as shown in Figure 6.1.

Given the current APR, the algorithm finds the sample from each segment that is closest to the APR using the Manhatten distance. Then, the APR is expanded to include the closest of those samples. Given the current set of segments that are covered by the APR, the algorithm checks to see if selecting a different sample from those segments would make the APR smaller. This process iterates until all segments are covered by this "tight" APR.

If the resulting tight APR does not contain a negative instance, the ID-APR uses a greedy method to find a subset of the features that are able to exclude all negative instances. A new tight APR is then found using this subset of features. On the PD dataset, however, the first APR always contained at least one negative instance. Therefore, the set of features was never reduced.

Because the tight APR needs only to cover one positive instance in each segment, it may be so small that none of the test samples are contained within. The authors of [34] therefore expand the APR such that, with high probability, new positive instances will lie within the APR. To do so, the authors use kernel density estimation to estimate the distribution of all samples from positive segments that are contained in the APR. Then the APR is expanded to contain ε of the resulting probability distribution. The kernel width is chosen such that, if the kernel function were centered within the APR, τ of the probability would lay within



Feature vectors from segment 2
a_y
b_y
Minimax APR
Feature vectors from segment 3
y
x
Feature vectors from segment 1
b_x
a_x

Figure 6.1: Example of minimax APR

the APR bounds. In [34], the authors report performance for $\tau = 0.99, 0.995$, and 0.999 and found the best performance was obtained for $\tau = 0.999$. For all experiments presented in this chapter, τ was fixed to be 0.999. Model selection, details given in Section 6.2, was used to choose the value of ε. For more details on the ID-APR algorithm, I refer readers to [34].

6.1.4 Expectation Maximization Diverse Density (EM-DD)

This algorithm was developed by Zhang et al. [139] as an extension of the Diverse Density (DD) algorithm developed by Maron and Lozano-Perez [87]. DD is a measure of how many *different* positive segments have samples near a particular point and how far negative samples are from that point. Distance is measured using a weighted Euclidean distance. The goal is therefore to find a set of weights and a point in feature space that maximizes DD. Maron and Lozano-Perez optimized this function using gradient ascent. To increase the chance of finding the global optimum instead of getting stuck in local maxima, the authors chose to initialize gradient ascent from multiple starting points. In particular, because DD measures the closeness of positive samples, at least one of the samples from a positive segment must be close to the global maximum. Therefore, initializing gradient ascent from each of the samples in positive segments should lead to the global maximum.

To reduce computation time, Zhang et al. [139] chose an expectation-maximization approach. Rather than using all samples from all positive segments to compute DD, they choose one sample from each positive segment at each step. In the E step, the current hypothesis is used to select which sample from each positive segment is most likely to be the positive one. In the M step, only these selected samples are used to compute the new hypothesis. I used the EM-DD implementation provided by the MIL Toolkit [135].

Table 6.1: Percentage of each type of segment label for varying segment lengths

Segment length	Standard labels Positive	Standard labels Negative	Stratified labels 0-33%	Stratified labels 33-66%	Stratified labels 66-100%
30 s	59.8	40.2	61.2	14.5	24.3
1 min	70.3	29.7	61.4	17.2	21.4
3 min	85.5	14.5	61.9	19.2	18.8
5 min	89.3	10.7	61.7	20.5	17.8
10 min	93.8	6.3	59.0	24.3	16.7

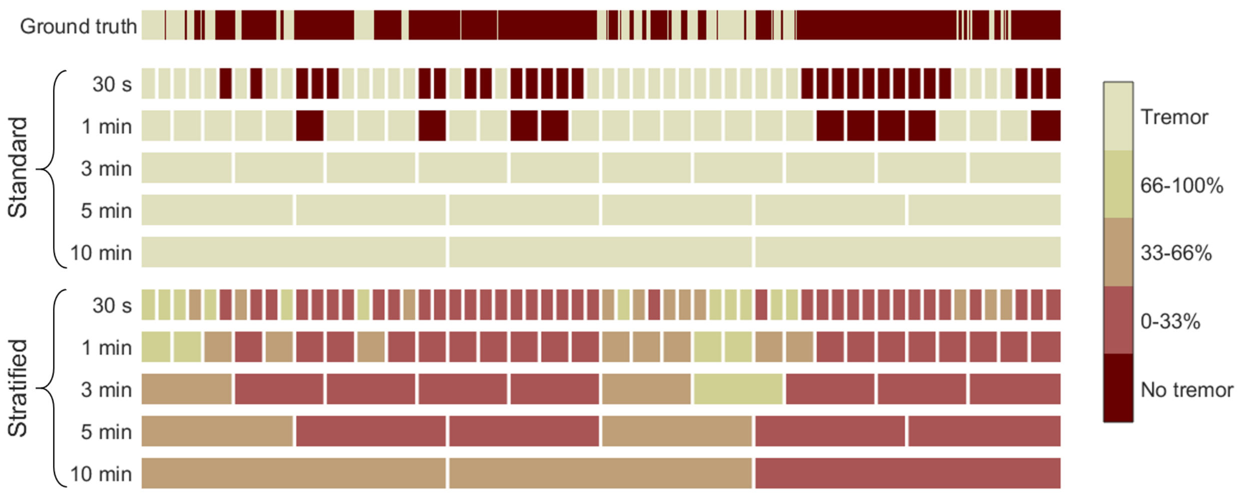


Figure 6.2: Example ground truth (30 minutes) and associated standard or stratified labels

6.1.5 Stratified Multiple Instance Learning

All algorithms described thus far use labels that only indicate whether a segment contains at least one positive sample or not. Performance of these algorithms could be improved if they had access to slightly more nuanced labels. For example, labels could contain the approximate percentage of tremor within a time segment (*e.g.*, 0-33%, 33-66%, 66-100%). In the scenario where participants are labeling their own in-home data, these stratified segment labels could correspond to labels of "almost no tremor," "about half," and "almost constant tremor." Figure 6.3a gives an example of standard versus stratified weak labels on samples in feature space.

In addition to containing more information, these stratified labels also have the advantage that they are symmetric between tremor and non-tremor. In contrast, with standard labels, most segments are labeled as containing tremor as segment length increases (see Figure 6.2) because increased segment length is associated with a higher likelihood of tremor occurring within. Very few negative examples can make it difficult for an algorithm to discriminate between tremor and non-tremor. The proportion of each type of stratified label, however, remains relatively constant as segment length increases (see Table 6.1).

MI-SVM and ID-APR are both solved by iterating between choosing a single event from each positive segment to serve as a positive selector variable (all events in negative segments are considered negative selector variables) and solving the decision boundary given the selector variables. Modifying these algorithms to use stratified labels simply involves changing the number of selector variables pulled from each bag. The number of positive and negative selector variables is adjusted according to the segment label. For example, given a segment with a label of 33-66%, we know that *at least* 33% of the segment must be tremor events, and *at least* 33% of the segment must be non-tremor events. Therefore, the highest scoring 33% of the events are chosen to be positive selector variables, and the lowest scoring 33% are chosen to be negative selector variables. The algorithm is then trained on these selector variables as normal. Figure 6.3d gives an example of how these selector variables are chosen. For a naive baseline using these labels, positive labels are assigned to segments with 66-100% tremor, negative labels to the segments with 0-33% tremor, and then the naive assumption is applied to the events in these segments and train a linear SVM.

I call these modified algorithms "stratified" and they are closely related to work on learning from label proportions (LLP) [57, 75, 106]. Much of the work in LLP involves new, probabilistic models that assume the exact proportions of positive to negative events are known. Kück and de Freitas allow for uncertainty in the proportions, but do so by introducing a user-specified parameter to represent this uncertainty [75]. In contrast, my solution is a simple modification of existing algorithms given *approximate* proportions of labels with no additional parameter.

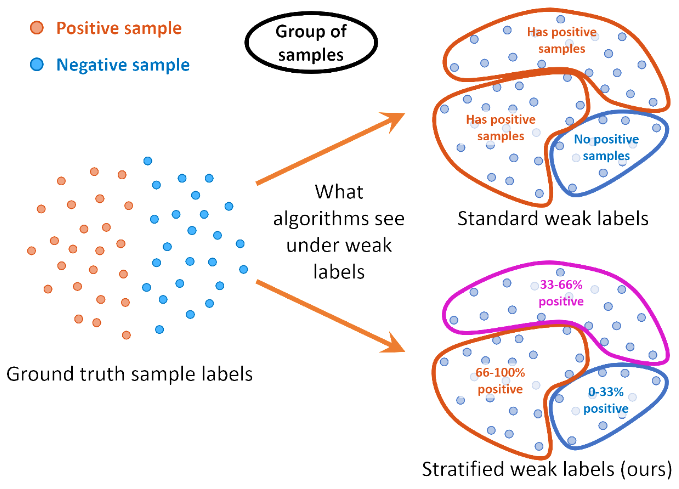
6.2 Methods

For all algorithms, I used the LAB dataset and features described in Chapter 3.1 and 3.4, respectively. Recall that features were computed over windows, and each window was labeled as tremor if at least half of the window was tremor. Consecutive windows were collected into segments of varying lengths (from 30 seconds to 10 minutes). Segments were given two different types of weak labels:

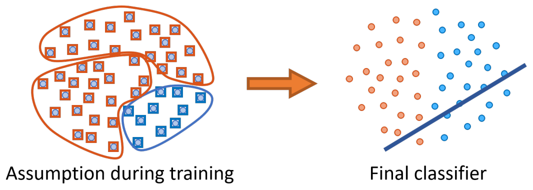
- Standard weak labels – positive if the segment contains at least one "tremor" window (event), negative otherwise.
- Stratified weak labels – approximate percentage of tremor (*e.g.*, 0-33%, 33-66%, 66-100%) witinh the segment.

As mentioned in Chapter 3.3, data were split into training and testing sets using leave-one-subject-out cross-validation: data from one subject were left out and the algorithm was trained on data from the remaining subjects. This process was repeated for each subject.

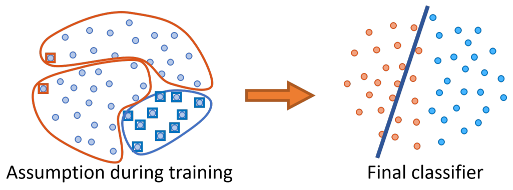
The SVM algorithms require the user-specified parameter C and the ID-APR algorithm uses the parameter ε. These parameters were selected automatically through model selection. During training, subjects in the training set were split into three groups, creating three folds



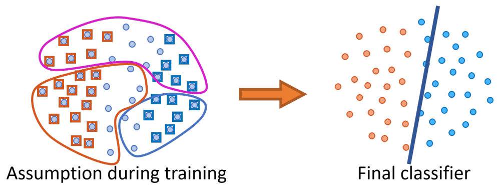
(a) Comparison of ground truth sample labels in feature space and associated standard or stratified weak labels



(b) Naive-SVM assignment of sample labels and final classifier



(c) MI-SVM assignment of sample labels and final classifier



(d) Stratified MI-SVM assignment of sample labels and final classifier

Figure 6.3: Example of standard and stratified weak labels in feature space and comparison of how Naive-SVM, MI-SVM, and Stratified MI-SVM learn from weak labels. Circles around samples indicate the set of all samples in feature space belonging to the same time segment. Colored squares around gray circles indicate labels assigned by the algorithm during training.

of the dataset. Models using each parameter value were trained on two folds and validated on the third fold. Using the same reasoning presented in Chapter 4.2.2, the set of C parameters to test was $C \in [2^{-13}, 2^{-12}, \dots, 2^{-1}]$. Following what was presented in Dietterich et al. [34], the set of ε parameters was chosen to be $\varepsilon \in [0.002, 0.004, \dots, 0.038, 0.040]$. This process was repeated for all three permutations of selecting two folds for training and one for validation. The parameter that resulted in the highest median performance across the three folds was used to train a model on the full training dataset. Note that performance during model selection was computed using segment-level labels as opposed to event-level labels because, in a real scenario, event-level labels would not be available to the algorithm. In particular, for non-stratified algorithms, performance was measured by segment-level label accuracy. For stratified algorithms, because the stratified labels are more nuanced, performance was measured by mean absolute error between output and labeled percentage of tremor (more details on this metric in Section 6.3.2).

6.3 Results and Discussion

The goal of this chapter is to explore how the performance of weakly supervised algorithms degrades as labels become less precise (time segments become longer). Results presented here helped inform the in-home data collection described in Chapter 3, where subjects assigned coarse labels (approximate amount of tremor) to short time segments throughout the day.

6.3.1 Standard Metrics

Algorithm performance is compared on five standard metrics – accuracy, F1-score, precision, recall, specificity, and AUC, described in Chapter 3.5. These performance metrics were computed on event-level labels (as opposed to segment-level labels) to measure whether the algorithms were able to discriminate tremor from non-tremor after being trained on the weakly labeled data.

Algorithm performance is summarized in Figure 6.4. We can see that the Naive-SVM algorithm fails once input segments reach a length of three minutes: specificity falls to zero, implying that the algorithm has converged to classifying everything as positive (tremor). This failure demonstrates the necessity of weakly supervised algorithms for accommodating weakly labeled data.

Performance of the standard weakly supervised algorithms are generally better than the Naive-SVM, particularly with respect to accuracy, precision, and specificity. Contrary to what Das et al. found [32], however, the performance of ID-APR is generally worse than that of MI-SVM and EM-DD. In fact, at 3-minute-length segments, its recall has fallen to nearly zero, implying that it is classifying nearly everything as non-tremor. Given that ID-APR is trying to find the smallest APR that includes at least one positive event, it is likely that the resulting APR for 3-minute segments is far too small, excluding the majority of tremor events.

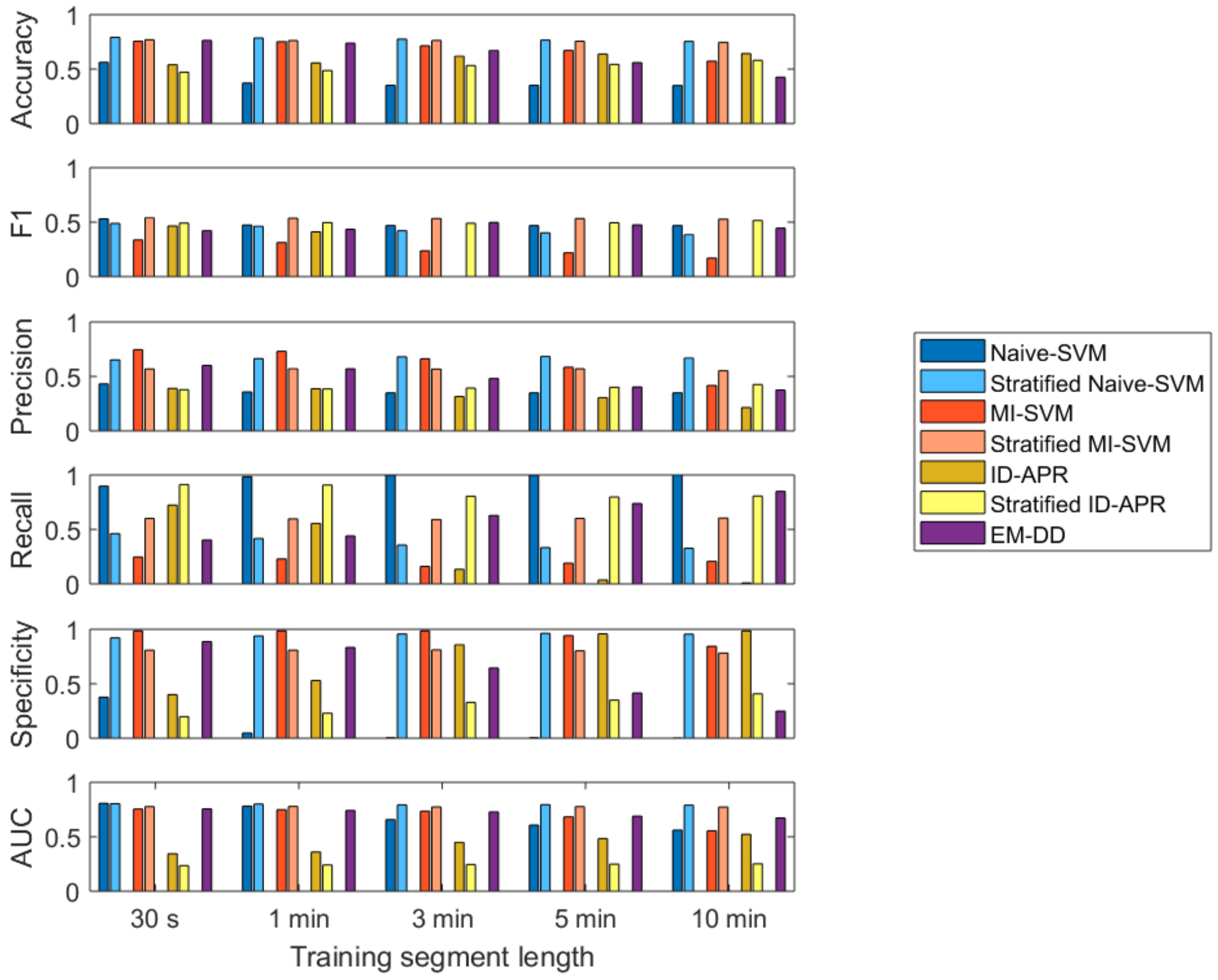


Figure 6.4: Performance comparison of all seven algorithms over varying lengths of training time segments using standard metrics.

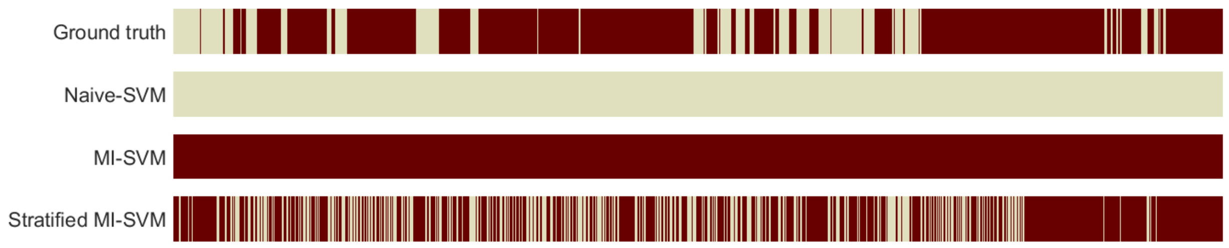


Figure 6.5: Empirical results of Naive-SVM, MI-SVM, and stratified MI-SVM on 30 minutes of ground truth data. Dark red indicates non-tremor.

As expected, performance of the non-stratified algorithms generally falls as the segment length increases because the labels become less precise. Meanwhile, the stratified algorithms are able to maintain consistent performance across all segment lengths. Naive-SVM and MI-SVM clearly benefit from stratification, with improved accuracy and AUC for all training segment lengths. Furthermore, stratification increases the F1 score for MI-SVM and only slightly decreases the F1 score of Naive-SVM. Stratification did not improve performance for the ID-APR algorithm, most likely because it requires many more samples to be included within the APR, which may make the APR too large.

For 10-minute-length segments, we see that all non-stratified algorithms have converged to classifying nearly everything as positive (specificity ≈ 0) or negative (recall ≈ 0). In contrast, the stratified algorithms (excluding the naive version) avoid this behavior. Figure 6.5 shows empirical results of Naive SVM, MI-SVM, and stratified MI-SVM on the same 30-minute segment shown in Figure 6.2. The stratified MI-SVM is better at capturing that this segment contains a mixture of tremor and non-tremor, and correctly shows a reduction in tremor near the end of the segment. Additional post-processing could help further refine the output of stratified MI-SVM. It is interesting to note that while stratified Naive-SVM and stratified MI-SVM have similar accuracy and AUC, stratified MI-SVM has higher F1 and recall, indicating that it is more able to find tremor. Using these metrics it is not clear which of the stratified methods should be preferred. However, the proposed performance metric (described below) is able to shed light on which is the superior algorithm.

6.3.2 Proposed Performance Metric

The above section reports performance along many metrics, but it is unclear which is most important to optimize. AUC has the advantage of being independent of class proportions, but it is the least interpretable. Meanwhile, accuracy is easily interpretable, but highly susceptible to being misleading due to class imbalance. All of these metrics measure how well an algorithm can classify a particular sample or instant of time. However, clinicians may find the amount of tremor that occurred within a given period of time to be more informative than whether a particular instant is tremor or not. Therefore, the error in the detected percentage of tremor within a window versus the true percentage of tremor may

Table 6.2: Mean absolute error of detected tremor percentage within 15-minute windows

	Training segment length				
Algorithm	**30 s**	**1 min**	**3 min**	**5 min**	**10 min**
Naive-SVM	39.5	63.3	66.0	66.0	66.2
MI-SVM	22.8	23.3	26.3	28.8	34.4
ID-APR	33.9	26.8	24.9	30.7	32.6
EM-DD	16.1	17.0	21.5	33.8	50.5
Naive-SVM stratified	14.8	16.1	18.0	19.5	20.4
MI-SVM stratified	**14.2**	**14.8**	**14.5**	**15.6**	**16.4**
ID-APR stratified	49.7	47.6	38.7	37.7	34.3

be a more clinically relevant performance metric. While it is true that the percentage of tremor can be computed from the detected instances of tremor, this metric is more forgiving with respect to errors in temporal shift of tremor events or mislabeling as long as the overall detected percentage remains the same. 15-minute windows are chosen because this resolution is short enough to give information about the effect of a patient's medication while remaining easily interpretable for clinicians.

Table 6.2 shows the mean absolute error (MAE) in detected tremor for all seven algorithms as they are trained on segments of varying length. We can see that the stratified MI-SVM shows the best performance in this metric across all segment lengths. Figure 6.6 rearranges the data from Table 6.2 to better show the effect of segment length and the stratified method on the MAE in detected tremor. For all non-stratified algorithms (excluding ID-APR), increased segment length leads to increased MAE. Meanwhile, stratified Naive-SVM and stratified MI-SVM are able to maintain performance and demonstrate a significant reduction in MAE compared to their non-stratified counterparts. ID-APR demonstrates curious behavior. The increase in MAE from stratification is likely due to the issue of the APR becoming too large, as discussed above. Interestingly, MAE decreases for stratified ID-APR as segment length increases, which may be due to increased freedom in choosing which samples from each segment would need to be included in the APR.

6.4 Conclusion

In this chapter, I simulated the types of labels that will be available from in-home data and compared the ability of several algorithms to learn from these weak labels. Using the accurately labeled LAB dataset, I could control the length of the training segments and thereby analyze how algorithm performance degraded as labels were made less precise. Furthermore, because ground truth labels exist for every time point, the ability of the algorithms to discriminate tremor from non-tremor events could be accurately evaluated. The work in this chapter thus serves to extend that of Das et al. [32], where only two subjects were used (each with a different motor symptom) and where ground truth labels were not available. This

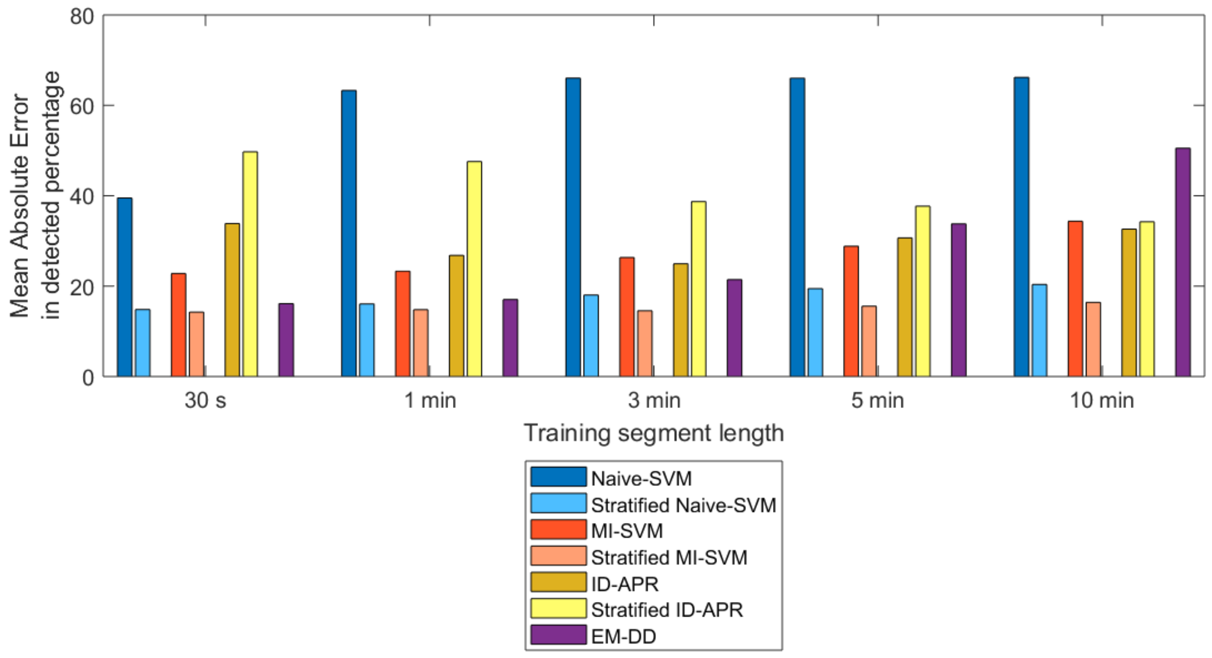


Figure 6.6: Effect of stratification on the mean absolute error of detected tremor percentage within 15-minute windows

chapter also describes a novel modification to MI-SVM and ID-APR, which allow them to take advantage of knowing the approximate amount of tremor within a given time segment.

The analyses in this chapter resulted in two main findings. (1) Contrary to that reported in Das et al. [32], ID-APR did not have the best performance. I believe my results differ due to the more thorough analysis of multiple performance metrics and the larger dataset. (2) My novel, stratified algorithms generally showed improved performance over their standard counterparts, particularly as the segment length increased. Results show that the stratified version of MI-SVM gave the best performance overall. Recall that all experiments presented in this chapter were conducted on laboratory data with weak labels computed from the accurate labels. In the next chapter, I apply stratified MI-SVM to in-home data and analyze how well laboratory data approximate in-home data.

Chapter 7

Learning from Wild Vs. Laboratory Data

This chapter analyzes the validity of the common assumption that training on laboratory data will lead to models that can generalize to in-the-wild data. Fully supervised algorithms were trained on laboratory data and weakly supervised algorithms on in-the-wild data. The performance of all algorithms was compared on both laboratory and in-the-wild data. These experiments measure how well laboratory data represents in-the-wild data. They also compare the efficacy of accurate labels, which are only available from laboratory data, to labels collected in-the-wild, which are necessarily weak and therefore less precise. Results demonstrate that training on laboratory data does not translate very well to the wild, which, surprisingly, leads the weak in-the-wild labels to be more effective than accurate laboratory labels.

7.1 Methods

To compare how well LAB data represents WILD data (data collection details described in Chapter 3), and to understand what kinds of training datasets would be most effective for an automated, in-home, tremor monitoring system, I partitioned the dataset into several different subsets for training Support Vector Machines (SVMs):

- Laboratory data from subjects who were not the test subject, which represents the standard leave-one-subject-out (LOSO) machine learning paradigm.
- Laboratory data from the test subject, which represents a person-specific model trained on data collected in controlled settings with accurate labels.
- In-the-wild data from the test subject, which represents a person-specific model trained on data collected in naturalistic settings with weak labels.

The three partitions are described in detail below.

The SVMs were implemented using the LIBLINEAR library provided by Fan et al. [37]. Features were those described in Chapter 3.4. Training data were normalized to have a mean of zero and a standard deviation of one. (Note that different algorithms used different partitions of the training data, and normalization parameters thus differed across the algorithms.) In all cases, model selection for the hyperparameter C was chosen through three-fold cross validation. That is, the training dataset was partitioned into three folds, two of which were used for training and one for validation. Thirteen models were trained (one for each C in the range of 2^{-13} to 2^{-1} in powers of 2) on the training folds and evaluated on the validation fold. This procedure was repeated for each of the three possible permutations of selecting training and validation sets from the three folds. The results were averaged across the three trials and the highest performing C was then used to train a new model on the entirety of the training data set (all three folds). Performance of this model was then evaluated on the test dataset, which was distinct from the dataset used for training.

7.1.1 Models

Below, I describe the three different partitions of the dataset for training and the algorithms applied to each partition. Each choice of training data led to a different model (classification hyperplane). A graphical representation of the partitions is shown in Figure 7.1. A summary is given in Table 7.1, which also includes the number of samples available for training and testing of each algorithm. Recall from Chapter 3.3 that data from separate hands are considered to be from separate subjects.

Generic SVM from LAB data (***Gen-LAB***)

I evaluated a standard, binary SVM on LAB data using leave-one-subject-out cross validation: *i.e.*, training on all subjects excluding one, and testing on that left out subject. For model selection, the three folds were chosen such that each fold contained data from one third of the training subjects. To select C, I chose that which led to the highest average Area Under the Curve (AUC) value across the three possible validation sets. The final model was then tested on the test subject's LAB and WILD data. This experimental protocol represents a typical machine learning pipeline and is subsequently referred to as *Gen-LAB*.

Person-specific SVM from LAB data (***PS-LAB***)

I trained a standard, binary SVM on LAB data from the test subject. The LAB dataset was first partitioned into three folds, two of which were used for training/validation and one for testing. Model selection was performed on the training/validation portion. The learned SVM was then tested on the test partition of the LAB data and the entire WILD dataset. Results were then averaged across all three learned SVMs, corresponding to three permutations of selecting the training/validation and test sets from the three folds of the LAB data. Following the recommendation of [49], all folds were chosen to be temporally connected segments, rather than selected from randomized samples. It has been shown that training a person-specific classifier leads to improved performance over a generic classifier [132], and I compared performance on LAB versus WILD data for such a classifier, which is subsequently referred to as *PS-LAB*.

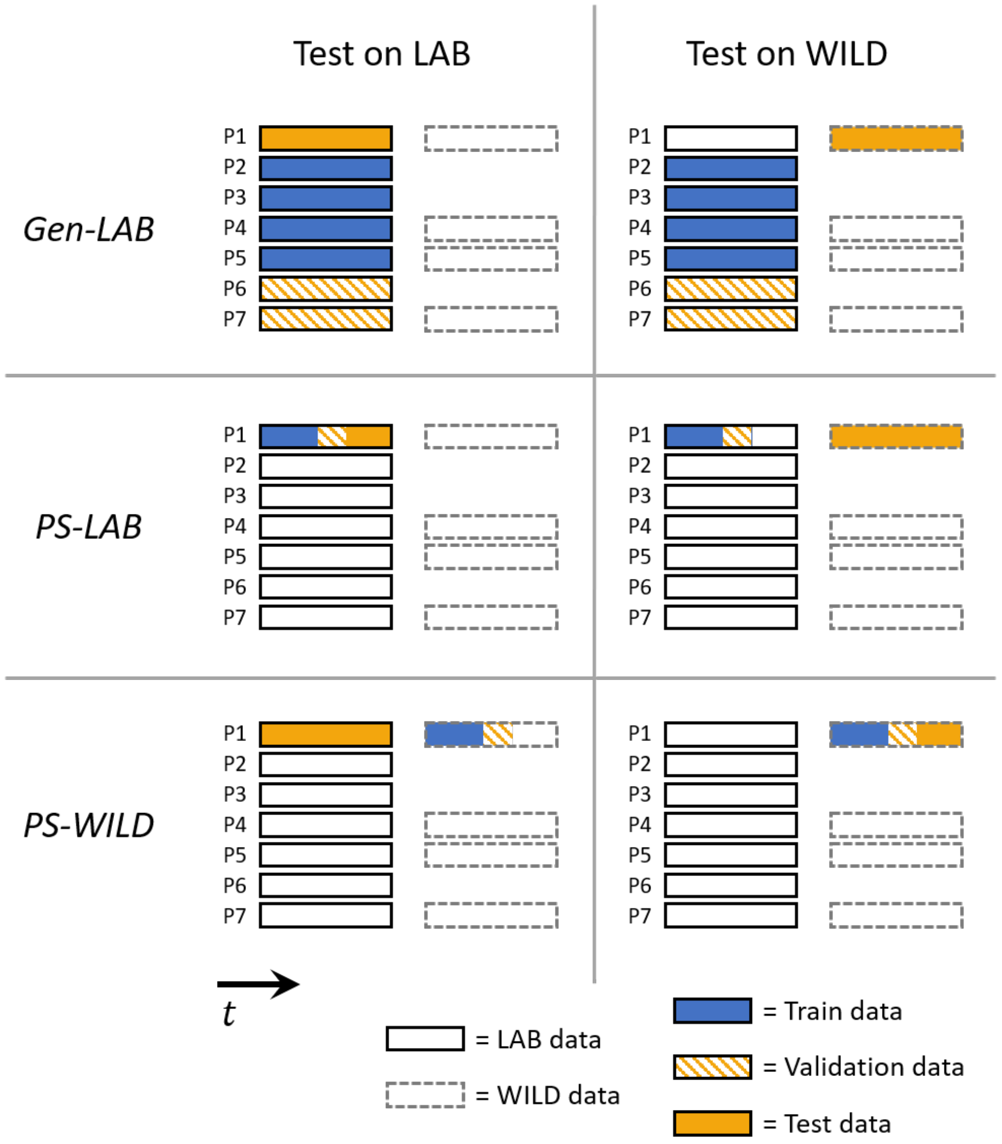


Figure 7.1: Schematic representation of training, validation, and test datasets for all experiments. Cross validation (averaging validation results across all folds) was used to do model selection. For *PS-LAB* and *PS-WILD*, the subject's LAB or WILD data were split into three folds, two of which were used for training/validation. Three models were trained, one for each permutation of selecting folds for training and testing. Results were averaged across all three models.

Table 7.1: Summary of experiments

	Training			**Test**		
Model	Dataset	Label type	Mean # labels / sub	Dataset	Label type	Mean # labels / sub
Gen-LAB	LAB (from others)	Accurate	83.7k (samples)	LAB	Accurate	4.3k (samples)
				WILD	Weak	388 (segments)
PS-LAB	⅔ LAB†	Accurate	2.9k (samples)	⅓ LAB‡	Accurate	1.4k (samples)
				WILD	Weak	388 (segments)
PS-WILD	⅔ WILD†	Weak	359 (segments)*	LAB	Accurate	4.3k (samples)
				⅓ WILD‡	Weak	129 (segments)

Note: excluding the training dataset for *Gen-LAB*, all datasets are from the test subject.
† The corresponding training dataset is partitioned into three folds while maintaining temporal consistency (*i.e.*, not randomized prior to splitting). Two folds are selected for training. There are three permutations of selecting two folds for training, therefore three different models are trained. Results from the three models are averaged.
‡ For these experiments, each model is tested *only* on the left out fold. Results are averaged across the three folds.
* WILD data have weak labels for 5 minute segments of time, each of which contain 328 samples. At every iteration of training the stratified MI-SVM algorithm, 66% of the samples from each segment are assigned labels according to the current hypothesis for the SVM decision boundary. Therefore, the stratified MI-SVM is trained on 216 samples per segment.

Person-specific SVM from WILD data (*PS-WILD*)

Using the WILD data from the test subject, I trained a stratified, Multiple Instance SVM (MI-SVM), as was used by Zhang et al. in [138], and described in the previous chapter in Section 6.1.5. Similarly to the methodology for *PS-LAB*, the WILD dataset was first partitioned into three folds (two for training/validation and one for testing). The learned SVMs were tested on the test partition of the WILD dataset and the entirety of the LAB dataset. Results were averaged across all three permutations. As with *PS-LAB*, folds were chosen to be temporally connected. As shown in Figure 3.5, labeling frequency was relatively consistent across the data collection period. Therefore, these partitions correspond to similar numbers of days. Because accurate labels are not available for WILD data, AUC could not be used for selecting C. Instead, I chose to use mean absolute error (described below in Section 7.1.2) as the performance metric for model selection. Note that, for some partitions, training data would lack segments with "*Almost none*" or "*Almost always*" labels, making it not possible to initialize the stratified MI-SVM algorithm with the method described above. Such partitions were ignored during model selection.

Stratified MI-SVM was trained as follows: Labels of *Almost none*, *Half the time*, and *Almost always* were assigned approximate tremor percentages of [0-33%], [33-66%], and [66-100%], respectively. To initialize the stratified MI-SVM algorithm, I computed the mean of each segment either labeled as *Almost none* or *Almost always*. These means were given negative and positive labels, respectively, and used to train the initial SVM. Training proceeded as described in Chapter 6.1.5. Termination conditions were set to be either when the set of selected samples is no longer changing, or when the maximum number of iterations (set to be 200) is reached. Note that, if the algorithm chooses a set of samples that has already been chosen, it enters an infinite loop. Therefore, a third termination condition was if the set of samples for the next iteration has been chosen before. In practice, depending on the dataset and chosen parameters, any of these termination conditions could occur.

The purpose of training on WILD data is to explore the relative benefits of accurate labels from other subjects versus the less precise weak labels from the test subject. This classifier is subsequently referred to as *PS-WILD*.

7.1.2 Performance metrics

Three performance metrics were used to evaluate the *Gen-LAB*, *PS-LAB*, and *PS-WILD* models:

- Accuracy (see Chapter 3.5).
- Area under the (Receiver Operating Characteristic) curve (AUC) (see Chapter 3.5).
- Mean absolute error in detected percentage: for each segment, the percentage of tremor within was computed using the trained model. If the detected percentage was within the associated range for the given label, the error on that segment was set to be zero. Otherwise, it was set to be the absolute difference between the detected percentage and the closest bound. For example, if a segment's label was *Almost always* (associated range of [66-100%]) and the detected percentage of tremor in that segment was 50%,

then the absolute error on that segment would be 16%. The absolute error was then averaged across all segments to get the mean absolute error.

Note that, because accurate labels (exact time points of tremor events) are not available for WILD data, I could not compute accuracy and AUC on WILD data. To compute mean absolute error in detected percentage on LAB data, I synthetically generated weak labels on non-overlapping 5-minute segments. Using the accurate labels, I determined the percentage of tremor within each segment. Given the percentage of tremor, I assigned weak labels consistent with how I had assigned tremor percentage ranges to the WILD labels: tremor percentages within [0-33%], [33-66%], or [66-100%] were assigned labels of *Almost none*, *Half the time*, or *Almost always*, respectively.

7.2 Results and Discussion

From Figure 3.5, we can see that participants 2 and 4 had highly variable tremor: almost every day, there were periods with "*Almost none*," "*Half the time*," or "*Almost always*" tremor. This variability makes them ideal candidates for an automated monitoring system. Furthermore, having a more even distribution of segments into the three types of labels facilitates machine learning. The remaining four participants had less variability in the presence/absence of tremor, which makes it more difficult to apply machine learning due to class imbalance. However, such participants may also benefit less from an automated symptom monitoring system: if a patient always or never experiences tremor, a monitoring system may not be needed. Note that, while patient 11 did experience variability *between* hands (left hand tremors almost constantly, and right hand has almost no tremor), I did not combine hands during training, as mentioned above, because such an algorithm would most likely only learn to differentiate left and right hands, rather than tremor versus no tremor. Participant 5 is also of interest: after the first week, almost all labels were "*Half the time*." While we did explain the importance of labeling accurately, it is unclear whether these labels are correct or due to misunderstanding. Nonetheless, results from all participants are included for completeness.

Table 7.2 shows accuracy and AUC values on LAB data for the *Gen-LAB*, *PS-LAB*, and *PS-WILD* classifiers. Consistent with previous findings on person specific classifiers [132], the *PS-LAB* classifier has the highest performance. The *Gen-LAB* classifier has slightly lower accuracy than *PS-LAB* on average, but very similar AUC values. Meanwhile, *PS-WILD* has much lower performance. These results are unsurprising for two reasons: (1) weak labels are inherently less precise than accurate labels, and (2) LAB and WILD data are poor representations of each other, as shown by results from Table 7.3 discussed below.

Table 7.3 compares mean absolute error values on LAB and WILD data for the three classifiers. We can see a large performance drop from LAB to WILD data: all mean absolute errors increase. The large variations in performance indicate that LAB data may not be very representative of WILD data. Interestingly, the largest performance deviation occurs with the *PS-LAB* classifier, implying that it is likely overfitting to the individual's LAB data. Consistent with findings from Hammerla [50], results show that the *PS-WILD* classifier

Table 7.2: Comparing accuracy and area under the curve (AUC) on LAB data (experimental setup shown in Figure 7.1, left column)

		Accuracy			**AUC**		
Subject	**Hand**	*Gen-LAB*	*PS-LAB*	*PS-WILD*	*Gen-LAB*	*PS-LAB*	*PS-WILD*
2	L	71.6	**89.9**	57.2	0.84	**0.86**	0.64
	R	79.9	**85.4**	29.1	0.85	**0.90**	0.65
4	L	**83.4**	79.5	39.6	**0.89**	0.80	0.72
	R	**89.5**	88.9	28.4	**0.92**	0.88	0.80
5	L	67.1	**74.6**	22.1	0.77	**0.80**	0.49
	R	63.4	**69.1**	20.7	0.76	**0.77**	0.54
10	L	86.5	**87.6**	3.2	0.75	**0.79**	0.56
	R	86.0	**87.4**	2.2	0.79	**0.88**	0.49
11	L	86.4	**88.1**	69.3	0.93	**0.94**	0.80
	R	83.2	**83.3**	0.0	0.88	**0.89**	0.75
12	L	87.3	**97.8**	0.3	**0.87**	0.77	0.53
	R	84.0	**98.7**	0.9	**0.79**	0.75	0.76
Average		80.7	**85.9**	22.7	**0.84**	**0.84**	0.64

Table 7.3: Comparing mean absolute error (MAE) in detected percentage on LAB and WILD data

		Test on LAB			**Test on WILD**		
Subject	**Hand**	*Gen-LAB*	*PS-LAB*	*PS-WILD*	*Gen-LAB*	*PS-LAB*	*PS-WILD*
2	L	8.90	**0.22**	7.53	9.36	12.55	**7.47**
	R	1.07	**0.17**	6.84	11.51	13.09	**8.51**
4	L	**1.39**	5.11	2.66	9.75	7.68	**6.05**
	R	**0.06**	0.58	3.80	10.90	10.56	**5.46**
5	L	12.67	**3.54**	5.93	14.37	11.16	**4.50**
	R	15.35	10.61	**9.39**	11.75	8.49	**3.80**
10	L	**1.71**	2.22	12.72	6.44	8.93	**3.62**
	R	**1.15**	1.18	1.44	5.51	7.51	**3.12**
11	L	1.93	**1.50**	-	14.48	**12.72**	-
	R	**3.67**	5.00	16.58	4.64	3.57	**0.00**
12	L	**0.00**	**0.00**	3.52	3.81	7.51	**3.23**
	R	0.43	**0.00**	8.12	6.53	14.66	**5.26**
Average		4.03	**2.51**	7.14	9.09	9.87	**4.64**

* indicates results were averaged over only two partition. In the third partition, stratified MI-SVM could not be initialized because the training set was lacking either "*Almost none*" or "*Almost always*" segments.
Note: for participant 11 (L) none of the training partitions included any "*Almost none*" segments.

experiences the least variation in performance between LAB and WILD data. Furthermore, *PS-WILD* demonstrates the highest performance on WILD data. Figure 7.2 indicates that these results largely hold for each fold of the WILD data.

It is possible that one reason for the improved performance of *PS-WILD* is that it was able to learn the biases of the participants. That is, the participants may have perceived their tremors to occur more or less frequently than reality, and the algorithm learned to concur with these skewed perceptions. Alternatively, they may have interpreted the labels differently, and the algorithm learned each participant's particular interpretation. However, while it would certainly be beneficial to clinicians to have completely unbiased symptom monitoring, the *PS-WILD* is no more biased than the current gold standard, which is patient self-reports. Furthermore, *PS-WILD* can at least help increase the frequency of monitoring by automating it.

Another possible cause for the high performance of *PS-WILD* on the WILD data is due to overfitting. As described in [49], refraining from randomizing the samples before splitting into folds helps prevent performance estimates from being overly optimistic. However, to truly test generalizability, one would need to examine performance on data collected several months later. Nonetheless, these results indicate that training on 2/3 of weakly labeled WILD data generalizes better to the left out 1/3 than training on accurately labeled LAB data (as shown in Figure 7.2). It is possible that a laboratory dataset with many more participants and a broader set of activities could lead to better performance than the PS-WILD models. However, these findings suggests that, when building a system for automated, passive, continuous symptom monitoring, it may be more beneficial to personally tailor the system to specific users by training on their own, in-home, weakly labeled data than to invest significant resources in building a large, accurately labeled dataset from laboratory recordings of other people.

7.3 Conclusion and Future Work

In this chapter, I directly analyze how well data collected in laboratory settings represents data collected in-the-wild for the purpose of continuous, automated PD tremor detection. Previous work has typically trained machine learning algorithms on laboratory data under the assumption that results will generalize to in-the-wild data. Other work has collected data in-the-wild, but not collected labels for training the algorithms and assessing symptom detection performance on such data. Three different methods of partitioning the dataset were used to build three models – *Gen-LAB*, *PS-LAB*, and *PS-WILD* – per subject. For every model, performance on laboratory data differs greatly from that on wild data (see Table 6.2). Furthermore, while the person-specific classifier trained on LAB data (*PS-LAB*) has the highest performance on LAB data, it has the lowest performance on WILD data. These findings imply that we should not assume in-lab performance will transfer to the wild.

Another interesting finding is that the person-specific classifier trained on WILD data (*PS-WILD*) performs better on WILD data than either of the classifiers trained on LAB data (*Gen-LAB* and *PS-LAB*). It is expected that training an algorithm on data from a specific

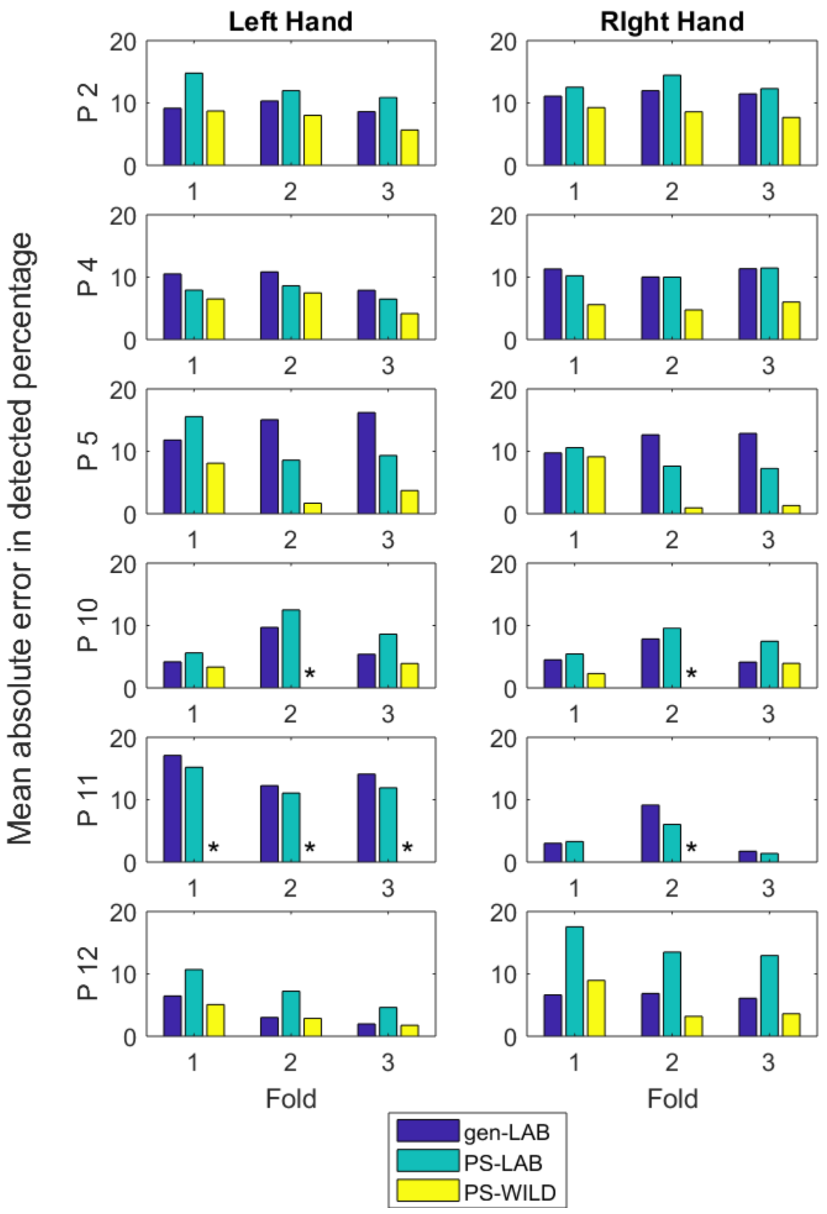


Figure 7.2: Absolute error of *Gen-LAB*, *PS-LAB* and *PS-WILD* for each of the three folds of WILD. Note that for *Gen-LAB* and *PS-LAB*, the same model is used for testing on each of the WILD folds, whereas for *PS-WILD*, the training set differs with each WILD test fold. * indicates that the training set was lacking "*Almost none*" or "*Almost always*" segments, which made it not possible to initialize the stratified MI-SVM algorithm.

user/environment will lead to higher performance on that user/environment. However, it is surprising that training on weak labels from the test subject, which contain less information, can outperform training on accurate labels from the test subject.

Together, these findings suggest that when developing a system for continuous, automated symptom detection, higher accuracy can be achieved by asking users to weakly label their own, in-the-wild data for training than to invest significant resources in building a training dataset collected from other people. In this way, machine learning algorithms can be tailored to each user and a person-specific baseline can be established for later comparison during monitoring.

The work in this chapter serves as a preliminary exploration into LAB versus WILD data. I envision a system where users might submit labels over the course of one or two weeks, after which a model would be trained and symptom detection would proceed automatically. It would be interesting explore how many labels users would need to provide for performance to stabilize, and whether the distribution of these labels over time affects performance. Before building a product requiring users to submit their own labels, future work should investigate whether these findings hold with other feature sets, datasets, and algorithms.

Chapter 8

Conclusion

In-home, continuous PD symptom monitoring could revolutionize PD patient care. This technology would help clinicians understand exactly how their patients' symptoms respond to medication; it would help patients understand how factors, such as sleep, stress, or food, affect their symptoms; and it would enable fine-grained, objective data collection on the effects of new treatments to prevent, slow the progression of, or cure PD. This thesis explores the two main challenges of developing this technology: (1) humans are highly variable, which makes it difficult for machine learning algorithms using wearable accelerometers to perform well on people who were not in the training set, and (2) it is difficult to obtain labels in the wild, which makes it difficult to train machine learning algorithms that will perform well in the wild.

To tackle the issue of human variability, many have formulated the problem as machine learning under covariate shift, *i.e.*, training and test data come from different distributions. Given unlabeled test data, the distribution of the test data can be estimated and leveraged to improve the classifier. This thesis explores several algorithms – Kernel Mean Matching (KMM) [45], Selective Transfer Machine (STM) [25], and Transductive Parameter Transfer (TPT) [114] – that have been developed to address the covariate shift problem. In particular, STM and TPT have been shown to perform well in human facial expression analysis from video data. Interestingly, in our dataset of PD tremor, none of the three algorithms were able to demonstrate significant improvement over a generic SVM.

To better understand the lack of improvement from STM, this thesis presents in-depth experiments on several synthetic datasets and a human activity recognition dataset. A multiclass version of STM was developed to accommodate these multiclass applications. Based on observations of performance in the synthetic datasets, this thesis also presents two modifications to the STM objective function that are meant to better describe the desired separating hyperplane. These modifications demonstrate improvement on the synthetic datasets. However, results are not as positive on the human activity recognition dataset. Further experiments show that, on these datasets, the STM objective function is not monotonically related to accuracy. Therefore, minimizing the objective function does not necessarily maximize accuracy. It would be interesting to conduct further experiments to understand the conditions necessary for these personalization algorithms to perform well.

This thesis also explores how to effectively train machine learning algorithms to perform well in the wild. A popular paradigm is to collect data in controlled laboratory conditions where video data would be available for labeling. These data are used for training and validation, and it is assumed that the learned models will perform similarly in the wild. Nowadays, with smartwatches and smartphones becoming more ubiquitous, it is easier to collect data in the wild. However, the labeling problem remains.

Accurate labels (the exact start and end of each event) are not feasible for in-the-wild data. However, weak labels (the approximate time of an event) are realistic. In this thesis, laboratory data were used to compute labels of varying levels of weakness (*i.e.*, labels over time segments of varying lengths) to explore how the performance of several algorithms is affected as labels become weaker. Additionally, this thesis presents a novel weakly supervised learning algorithm, called Stratified Multiple Instance Learning, which can be paired with many different classification algorithms. The assumptions of this algorithm align better with the reality of PD symptom detection: that symptoms can occur with any frequency, ranging from not at all to continuously. In contrast, other weakly supervised algorithms assume positive events occur very infrequently. Experiments demonstrated that standard multiple instance algorithms struggled as labels were made weaker, but the stratified versions were able to maintain their performance.

This thesis then applied the stratified algorithm to in-the-wild data. Experiments were conducted to analyze whether how well laboratory data represents in-the-wild data, and whether algorithms trained on lab data could perform well on wild data and vice versa. Results demonstrated that, for all algorithms, performance differed significantly between lab and wild data. These findings imply that lab data are *not* a good representation of wild data, and that researchers should not assume that performance on lab data will translate to the wild. All things being equal, training on wild data should perform better on wild data than training on lab data. However, things are not equal: labels on wild data are weak, whereas accurate labels are available for lab data. Interestingly, experiments show that the less precise, weak labels from wild data are still more effective for in-the-wild detection than accurate labels from lab data.

We hope that the technology presented in this thesis could be transferred to a product for automated symptom monitoring. Such a product would be worn continuously and log symptoms throughout the day. It would then present symptom prevalence over time to both the user and their doctor. A new data collection, incorporating the changes suggested in Chapter 3.7, would help ensure that the algorithms are robust across users and across time. To be truly useful for all PD patients, such a product would need to detect other motor symptoms, like bradykinesia, dyskinesia, and freezing of gait. Additional data collections would be required to build algorithms for monitoring these symptoms and to demonstrate that accelerometers are an effective sensing modality for these symptoms.

If the findings in this thesis – that weak labels from the wild are more effective than accurate labels from the lab – hold across larger datasets and other symptoms, then a product should require users to submit labels of their own data. We envision a system where users might submit labels over the course of one or two weeks, after which a model would be trained and symptom detection would proceed automatically. This product would require a well-designed user interface to make submitting labels as easy as possible. To improve algorithm performance, users could opt in to contributing their labels to a larger dataset for training algorithms on WILD data of numerous users. Additionally, the algorithm could choose to only classify samples for which it has high confidence, thereby improving classification accuracy at the cost of leaving some segments of time unclassified. Detected symptoms would need to be summarized in a clean and easily digestible format for both clinicians and patients. Ideally, these data would be sent to a clinician for review and archived in a patient's medical chart.

In future work, it would be interesting to see how many labeled samples or weeks of data are required for algorithm performance to stabilize. Additionally, it is possible that active learning could be used to help maintain high performance over longer periods of time with relatively minimal user effort. Another way to reduce labeling load of users is to apply personalization to the weakly labeled data. Importance reweighting algorithms are designed to work without labels from the end user. While those algorithms did not show large improvements on our dataset, it is possible that they could help improve performance on the weakly labeled data. Furthermore, these algorithms could be used in combination with a small number of labels from the end user. An extension of STM doing just that has already been proposed [25], and quite a few other solutions have been proposed, as described in Chapter 2.2.3. As wearable devices become more ubiquitous, avenues for nearly unsupervised learning become more feasible: with enough users, broad labels of healthy versus PD may be sufficient for an algorithm to learn to detect PD symptoms.

While this thesis focuses on PD symptom detection, the findings within should generalize to other problems in human activity understanding, such as monitoring of other motor impairments, activity tracking, or sports performance analysis. In these fields, it is often difficult to obtain sufficiently large, fully supervised datasets because assigning labels (*e.g.*, marking the beginning and end of each action in video data) is a highly labor-intensive and time-consuming task. Furthermore, labeling human behavior is ambiguous, uncertain, and labeler dependent. Weakly labeled data are often far easier to obtain. For instance, in our case, it is much easier to label a large segment of time in which a Parkinson's symptom occurred rather than precisely labeling the start and end of the motor symptom. Weakly supervised learning offers a new paradigm for building systems that can explicitly account for these less precise labels when training classifiers. Furthermore, the stratified version developed in this thesis should improve results for any application where the event of interest can occur over longer periods of time as opposed to in discrete instances.

Bibliography

[1] M Reyes Adame, A Al-Jawad, M Romanovas, MA Hobert, W Maetzler, K Möller, and Y Manoli. TUG test instrumentation for Parkinson's disease patients using inertial sensors and dynamic time warping. *Biomedical Engineering/Biomedizinische Technik*, 57(SI-1 Track-E):1071–1074, 2012. 2.3, 2.2

[2] Jake K Aggarwal and Michael S Ryoo. Human activity analysis: A review. *ACM Computing Surveys (CSUR)*, 43(3):16, 2011. 2.1

[3] Jake K Aggarwal and Lu Xia. Human activity recognition from 3d data: A review. *Pattern Recognition Letters*, 48:70–80, 2014. 2.1

[4] Saad Ali and Mubarak Shah. Human action recognition in videos using kinematic features and multiple instance learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(2):288–303, 2008. 2.3.3

[5] Jaume Amores. Multiple instance classification: Review, taxonomy and comparative study. *Artificial Intelligence*, 201:81–105, 2013. 2.3.3, 6

[6] Stuart Andrews, Ioannis Tsochantaridis, and Thomas Hofmann. Support vector machines for multiple-instance learning. *Advances in Neural Information Processing Systems*, pages 577–584, 2003. 6.1, 6.1.2

[7] S Arora, V Venkataraman, A Zhan, S Donohue, KM Biglan, ER Dorsey, and MA Little. Detecting and monitoring the symptoms of Parkinson's disease using smartphones: A pilot study. *Parkinsonism & Related Disorders*, 21(6):650–653, 2015. 2.3, 2.2, 2.3, 2.3.1

[8] Siddharth Arora, Vinayak Venkataraman, Sean Donohue, Kevin M Biglan, Earl R Dorsey, and Max A Little. High accuracy discrimination of Parkinson's disease participants from healthy controls using smartphones. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 3641–3644. IEEE, 2014. 2.3, 2.2

[9] Teresa Arroyo-Gallego, María Jesus Ledesma-Carbayo, Álvaro Sánchez-Ferro, Ian Butterworth, Carlos S Mendoza, Michele Matarazzo, Paloma Montero, Roberto López-Blanco, Veronica Puertas-Martin, Rocio Trincado, et al. Detection of motor impairment in Parkinson's disease via mobile touchscreen typing. *IEEE Transactions on Biomedical Engineering*, 64(9):1994–2002, 2017. 2.3, 2.2

[10] Marc Bachlin, Meir Plotnik, Daniel Roggen, Inbal Maidan, Jeffrey M Hausdorff, Nir Giladi, and Gerhard Troster. Wearable assistant for Parkinson's disease patients

with the freezing of gait symptom. *IEEE Transactions on Information Technology in Biomedicine*, 14(2):436–446, 2010. 2.4

[11] Zahari Abu Bakar, Nooritawati Md Tahir, and Ihsan M Yassin. Classification of Parkinson's disease based on multilayer perceptrons neural network. In *Signal Processing and Its Applications (CSPA), 2010 6th International Colloquium on*, pages 1–4. IEEE, 2010. 2.3, 2.2

[12] Oresti Baños, Miguel Damas, Héctor Pomares, Ignacio Rojas, Máté Attila Tóth, and Oliver Amft. A benchmark dataset to evaluate sensor displacement in activity recognition. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pages 1026–1035. ACM, 2012. 5, 5.2

[13] Oresti Banos, Mate Attila Toth, Miguel Damas, Hector Pomares, and Ignacio Rojas. Dealing with the effects of sensor displacement in wearable activity recognition. *Sensors*, 14(6):9995–10023, 2014. 5, 5.2

[14] Ling Bao and Stephen S Intille. Activity recognition from user-annotated acceleration data. In *International Conference on Pervasive Computing*, pages 1–17. Springer, 2004. 2.1

[15] Paulo Barbosa, Kemilly Dearo Garcia, João Mendes-Moreira, and André CPLF de Carvalho. Unsupervised domain adaptation for human activity recognition. In *International Conference on Intelligent Data Engineering and Automated Learning*, pages 623–630. Springer, 2018. 2.2.3

[16] Jens Barth, Jochen Klucken, Patrick Kugler, Thomas Kammerer, Ralph Steidl, Jürgen Winkler, Joachim Hornegger, and Björn Eskofier. Biometric and mobile gait analysis for early diagnosis and therapy monitoring in Parkinson's disease. In *Engineering in Medicine and Biology Society (EMBC), 2011 Annual International Conference of the IEEE*, pages 868–871. IEEE, 2011. 2.3, 2.2

[17] Gabriele Bleser, Daniel Steffen, Attila Reiss, Markus Weber, Gustaf Hendeby, and Laetitia Fradet. Personalized physical activity monitoring using wearable sensors. In *Smart Health*, pages 99–124. Springer, 2015. 2.2.3

[18] Brian M Bot, Christine Suver, Elias Chaibub Neto, Michael Kellen, Arno Klein, Christopher Bare, Megan Doerr, Abhishek Pratap, John Wilbanks, E Ray Dorsey, et al. The mpower study, Parkinson disease mobile data collected using researchkit. *Scientific Data*, 3:160011, 2016. 2.3.1, 2.5

[19] Bambi R Brewer, Sujata Pradhan, George Carvell, and Anthony Delitto. Application of modified regression techniques to a quantitative assessment for the motor signs of Parkinson's disease. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 17(6):568–575, 2009. 2.3, 2.3

[20] BR Brewer, S Pradhan, G Carvell, and A Delitto. Feature selection for classification based on fine motor signs of Parkinson's disease. In *Engineering in Medicine and Biology Society (EMBC), 2009 Annual International Conference of the IEEE*, pages 214–217. IEEE, 2009. 2.3, 2.2

[21] LM Chahine, L Uribe, P Hogarth, James McNames, A Siderowf, K Marek, and D Jennings. Portable objective assessment of upper extremity motor function in Parkinson's disease. *Parkinsonism & Related Disorders*, 43:61–66, 2017. 2.3.1

[22] Olivier Chapelle. Training a support vector machine in the primal. *Neural Computation*, 19(5):1155–1178, 2007. 5.1.3

[23] Yen-Ping Chen, Jhun-Ying Yang, Shun-Nan Liou, Gwo-Yun Lee, and Jeen-Shing Wang. Online classifier construction algorithm for human activity detection using a tri-axial accelerometer. *Applied Mathematics and Computation*, 205(2):849–860, 2008. 2.1

[24] Wen-Sheng Chu, Fernando De la Torre, and Jeffery F Cohn. Selective transfer machine for personalized facial action unit detection. In *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3515–3522, 2013. 4, 4.1.3, 4.1.3, 5.3

[25] Wen-Sheng Chu, Fernando De la Torre, and Jeffrey F Cohn. Selective transfer machine for personalized facial expression analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(3):529–545, 2016. 1, 2.2.2, 5, 5.7, 8

[26] Bryan T Cole, Serge H Roy, Carlo J De Luca, and S Hamid Nawab. Dynamic neural network detection of tremor and dyskinesia from wearable sensor data. In *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, pages 6062–6065. IEEE, 2010. 2.4, 2.3.1

[27] Bryan T Cole, Serge H Roy, Carlo J De Luca, and S Hamid Nawab. Dynamical learning and tracking of tremor and dyskinesia from wearable sensors. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 22(5):982–991, 2014. 2.4, 2.3.1, 3.4

[28] 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. 2.2

[29] Corinna Cortes, Mehryar Mohri, and Andrés Munoz Medina. Adaptation based on generalized discrepancy. *Journal of Machine Learning Research*, 20(1):1–30, 2019. 2.2.2

[30] Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2(Dec):265–292, 2001. 5.1.2

[31] Bozidara Cvetkovic, B Kaluza, M Luštrek, and Matjaz Gams. Semi-supervised learning for adaptation of human activity recognition classifier to the user. In *Proc. of Workshop on Space, Time and Ambient Intelligence, IJCAI*, pages 24–29. Citeseer, 2011. 2.2.3

[32] Samarjit Das, Breogan Amoedo, Fernando De la Torre, and Jessica Hodgins. Detecting Parkinsons' symptoms in uncontrolled home environments: a multiple instance learning approach. In *Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE*, pages 3688–3691. IEEE, 2012. 2.3.3, 3.2, 3.7, 6, 6.1, 6.3.1, 6.4

[33] Hal Daumé III. Frustratingly easy domain adaptation. *arXiv preprint arXiv:0907.1815*, 2009. 2.2.1

[34] Thomas G Dietterich, Richard H Lathrop, and Tomás Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1): 31–71, 1997. 6.1, 6.1.3, 6.1.3, 6.2

[35] Ürün Dogan, Tobias Glasmachers, and Christian Igel. Fast training of multi-class support vector machines: Technical report no. 03/2011. Technical report, University of Copenhagen, 2011. 5.1.2

[36] ELAN. Elan – annotation software. Max Planck Institute for Psycholinguistics, The Language Archive, Nijmegen, The Netherlands. URL `https://tla.mpi.nl/tools/tla-tools/elan/`. Accessed: Nov. 17, 2017. 3.1

[37] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008. 4.1.1, 4.1.2, 4.1.3, 4.2.1, 6.1.1, 6.1.2, 7.1

[38] Y Rakhshani Fatmehsari and F Bahrami. Lempel-ziv complexity criteria for nonlinear analysis of gait in patients with Parkinson's disease. In *Biomedical Engineering (ICBME), 2011 18th Iranian Conference of*, pages 137–141. IEEE, 2011. 2.3, 2.2

[39] Basura Fernando, Amaury Habrard, Marc Sebban, and Tinne Tuytelaars. Subspace alignment for domain adaptation. *arXiv preprint arXiv:1409.5241*, 2014. 2.2.1, 2.2.3

[40] James M Fisher, Nils Y Hammerla, Thomas Ploetz, Peter Andras, Lynn Rochester, and Richard W Walker. Unsupervised home monitoring of Parkinson's disease motor symptoms using body-worn accelerometers. *Parkinsonism & Related Disorders*, 33: 44–50, 2016. 2.3.3, 3.7, 6, 6.1

[41] Joseph P. Giuffrida, David E. Riley, Brian N. Maddux, and Dustin A. Heldman. Clinically deployable kinesia™ technology for automated tremor assessment. *Movement Disorders*, 24(5):723–730, 2009. 2.3.2

[42] Christopher G Goetz, Barbara C Tilley, Stephanie R Shaftman, Glenn T Stebbins, Stanley Fahn, Pablo Martinez-Martin, Werner Poewe, Cristina Sampaio, Matthew B Stern, Richard Dodel, et al. Movement disorder society-sponsored revision of the Unified Parkinson's Disease Rating Scale (MDS-UPDRS): Scale presentation and clinimetric testing results. *Movement Disorders*, 23(15):2129–2170, 2008. 1, 2.1, 2.3, 3.1

[43] Boqing Gong, Yuan Shi, Fei Sha, and Kristen Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2066–2073. IEEE, 2012. 2.2.1

[44] Jochen Gorski, Frank Pfeuffer, and Kathrin Klamroth. Biconvex sets and optimization with biconvex functions: a survey and extensions. *Mathematical Methods of Operations Research*, 66(3):373–407, 2007. 5.6.1

[45] Arthur Gretton, Alex Smola, Jiayuan Huang, Marcel Schmittfull, Karsten Borgwardt, and Bernhard Schölkopf. Covariate shift by kernel mean matching. In Joaquin

Quionero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D Lawrence, editors, *Dataset Shift in Machine Learning*, chapter 8, pages 131–160. The MIT Press, 2009. 2.2.2, 2.2.3, 4, 4.1.2, 4.2.2, 4.4, 8

[46] Robert I Griffiths, Katya Kotschet, Sian Arfon, Zheng Ming Xu, William Johnson, John Drago, Andrew Evans, Peter Kempster, Sanjay Raghav, and Malcolm K Horne. Automated assessment of bradykinesia and dyskinesia in Parkinson's disease. *Journal of Parkinson's Disease*, 2(1):47–55, 2012. 2.3.2, 3.7

[47] Xinze Guan, Raviv Raich, and Weng-Keen Wong. Efficient multi-instance learning for activity recognition from time series data using an auto-regressive hidden markov model. In *International Conference on Machine Learning*, pages 2330–2339, 2016. 2.3.3

[48] Hirotaka Hachiya, Masashi Sugiyama, and Naonori Ueda. Importance-weighted least-squares probabilistic classifier for covariate shift adaptation with application to human activity recognition. *Neurocomputing*, 80:93–101, 2012. 2.2.2

[49] Nils Y Hammerla and Thomas Plötz. Let's (not) stick together: pairwise similarity biases cross-validation in activity recognition. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 1041–1051. ACM, 2015. 7.1.1, 7.2

[50] Nils Y. Hammerla, James M. Fisher, Peter Andras, Lynn Rochester, Richard Walker, and Thomas Plotz. PD disease state assessment in naturalistic environments using deep learning. In *Proc. of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 1742–1748. AAAI, 2015. ISBN 0-262-51129-0. 7.2

[51] AM Ardi Handojoseno, James M Shine, Tuan N Nguyen, Yvonne Tran, Simon JG Lewis, and Hung T Nguyen. The detection of freezing of gait in Parkinson's disease patients using eeg signals based on wavelet decomposition. In *Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE*, pages 69–72. IEEE, 2012. 2.4

[52] Ali Hassan, Robert Damper, and Mahesan Niranjan. On acoustic emotion recognition: compensating for covariate shift. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(7):1458–1468, 2013. 2.2.3

[53] Zhen-Yu He and Lian-Wen Jin. Activity recognition from acceleration data using ar model representation and svm. In *2008 International Conference on Machine Learning and Cybernetics*, volume 4, pages 2245–2250. IEEE, 2008. 2.1

[54] Zhenyu He and Lianwen Jin. Activity recognition from acceleration data based on discrete consine transform and svm. In *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*, pages 5041–5044. IEEE, 2009. 2.1

[55] Zhenyu He, Zhibin Liu, Lianwen Jin, Li-Xin Zhen, and Jian-Cheng Huang. Weightlessness feature — a novel feature for single tri-axial accelerometer based activity recognition. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1–4. IEEE, 2008. 2.1

[56] Dustin A Heldman, Alberto J Espay, Peter A LeWitt, and Joseph P Giuffrida. Clinician versus machine: reliability and responsiveness of motor endpoints in Parkinson's disease. *Parkinsonism & Related Disorders*, 20(6):590–595, 2014. 2.3, 2.3

[57] Jerónimo Hernández-González, Iñaki Inza, and Jose A Lozano. Learning bayesian network classifiers from label proportions. *Pattern Recognition*, 46(12):3425–3440, 2013. 6.1.5

[58] Jeffrey D Hoffman and James McNames. Objective measure of upper extremity motor impairment in Parkinson's disease with inertial sensors. In *Engineering in Medicine and Biology Society (EMBC), 2011 Annual International Conference of the IEEE*, pages 4378–4381. IEEE, 2011. 2.3, 2.2

[59] Jin-Hyuk Hong, Julian Ramos, and Anind K Dey. Toward personalized activity recognition systems with a semipopulation approach. *IEEE Transactions on Human-Machine Systems*, 46(1):101–112, 2015. 2.2.3

[60] Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, 2002. 5.1.2, 5.3

[61] Derek Hao Hu and Qiang Yang. Transfer learning for activity recognition via sensor mapping. *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, 22(3):1962, 2011. 2.2

[62] Ryan P Hubble, Geraldine A Naughton, Peter A Silburn, and Michael H Cole. Wearable sensor use for assessing standing balance and walking stability in people with Parkinson's disease: a systematic review. *PloS one*, 10(4):e0123705, 2015. 2.3

[63] Nazli Ikizler-Cinbis and Stan Sclaroff. Object, scene and actions: Combining multiple features for human action recognition. In *European Conference on Computer Vision*, pages 494–507. Springer, 2010. 2.3.3

[64] Ozlem Durmaz Incel, Mustafa Kose, and Cem Ersoy. A review and taxonomy of activity recognition on mobile phones. *BioNanoScience*, 3(2):145–171, 2013. 2.1

[65] Luciana C Jatoba, Ulrich Grossmann, Chistophe Kunze, Jorg Ottenbacher, and Wilhelm Stork. Context-aware mobile health monitoring: Evaluation of different pattern recognition methods for classification of physical activity. In *Engineering in Medicine and Biology Society (EMBC), 2008 Annual International Conference of the IEEE*, pages 5250–5253. IEEE, 2008. 2.1

[66] Jing Jiang. A literature survey on domain adaptation of statistical classifiers, 2008. URL `http://www.mysmu.edu/faculty/jingjiang/papers/da_survey.pdf`. 2.2

[67] Takafumi Kanamori, Shohei Hido, and Masashi Sugiyama. A least-squares approach to direct importance estimation. *Journal of Machine Learning Research*, 10(Jul):1391–1445, 2009. 2.2.2, 2.2.3

[68] Panagiotis Kassavetis, Tabish A Saifee, George Roussos, Loukas Drougkas, Maja Kojovic, John C Rothwell, Mark J Edwards, and Kailash P Bhatia. Developing a tool

for remote digital assessment of Parkinson's disease. *Movement Disorders Clinical Practice*, 3(1):59–64, 2016. 2.3, 2.3

[69] Shian-Ru Ke, Hoang Le Uyen Thuc, Yong-Jin Lee, Jenq-Neng Hwang, Jang-Hee Yoo, and Kyoung-Ho Choi. A review on video-based human activity recognition. *Computers*, 2(2):88–131, 2013. 2.1

[70] Adil Mehmood Khan, Young-Koo Lee, Sungyoung Y Lee, and Tae-Seong Kim. A triaxial accelerometer-based physical-activity recognition via augmented-signal features and a hierarchical recognizer. *IEEE Transactions on Information Technology in Biomedicine*, 14(5):1166–1172, 2010. 2.1

[71] Faisal M Khan, Michael Barnathan, Michael Montgomery, Stanely Myers, Lucien Côté, and Sheree Loftus. A wearable accelerometer system for unobtrusive monitoring of Parkinson's disease motor symptoms. In *Bioinformatics and Bioengineering (BIBE), 2014 IEEE International Conference on*, pages 120–125. IEEE, 2014. 2.4, 3.4

[72] Daniel Kifer, Shai Ben-David, and Johannes Gehrke. Detecting change in data streams. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases*, volume 30, pages 180–191. VLDB Endowment, 2004. 2.2.2

[73] Mustafa R Kılınç and Nikolaos V Sahinidis. Exploiting integrality in the global optimization of mixed-integer nonlinear programming problems with baron. *Optimization Methods and Software*, 33(3):540–562, 2018. 5.6.1

[74] Ken J Kubota, Jason A Chen, and Max A Little. Machine learning for large-scale wearable sensor data in Parkinson's Disease: Concepts, promises, pitfalls, and futures. *Movement Disorders*, 31(9):1314–1326, 2016. 2.3

[75] Hendrik Kuck and Nando de Freitas. Learning about individuals from group statistics. *arXiv preprint arXiv:1207.1393*, 2012. 6.1.5

[76] Oscar D Lara and Miguel A Labrador. A survey on human activity recognition using wearable sensors. *IEEE Communications Surveys & Tutorials*, 15(3):1192–1209, 2013. 2.1

[77] Chae Young Lee, Seong Jun Kang, Sang-Kyoon Hong, Hyeo-Il Ma, Unjoo Lee, and Yun Joong Kim. A validation study of a smartphone-based finger tapping application for quantitative assessment of bradykinesia in Parkinson's disease. *PloS one*, 11(7): e0158852, 2016. 2.3, 2.2, 2.3

[78] Will Lee, Andrew Evans, and David R Williams. Validation of a smartphone application measuring motor function in Parkinson's disease. *Journal of Parkinson's Disease*, 6(2):371–382, 2016. 2.3, 2.3

[79] Yoonkyung Lee, Yi Lin, and Grace Wahba. Multicategory support vector machines: Theory and application to the classification of microarray data and satellite radiance data. *Journal of the American Statistical Association*, 99(465):67–81, 2004. 5.1.2

[80] Florian Lipsmeier, Kirsten I Taylor, Timothy Kilchenmann, Detlef Wolf, Alf Scotland, Jens Schjodt-Eriksen, Wei-Yi Cheng, Ignacio Fernandez-Garcia, Juliane Siebourg-

Polster, Liping Jin, et al. Evaluation of smartphone-based testing to generate exploratory outcome measures in a phase 1 Parkinson's disease clinical trial. *Movement Disorders*, 33(8):1287–1297, 2018. 2.3.1, 2.3.2

[81] Max A Little, Patrick E McSharry, Eric J Hunter, Jennifer Spielman, Lorraine O Ramig, et al. Suitability of dysphonia measurements for telemonitoring of Parkinson's disease. *IEEE Transactions on Biomedical Engineering*, 56(4):1015–1022, 2009. 2.3, 2.2

[82] Jeffrey W Lockhart and Gary M Weiss. The benefits of personalized smartphone-based activity recognition models. In *Proceedings of the 2014 SIAM International Conference on Data Mining*, pages 614–622. SIAM, 2014. 2.2.3

[83] Marco Loog. Nearest neighbor-based importance weighting. In *2012 IEEE International Workshop on Machine Learning for Signal Processing*, pages 1–6. IEEE, 2012. 2.2.2, 2.2.3

[84] Hany Hazfiza Manap, Nooritawati Md Tahir, and Ahmad Ihsan M Yassin. Statistical analysis of Parkinson disease gait classification using artificial neural network. In *Signal Processing and Information Technology (ISSPIT), 2011 IEEE International Symposium on*, pages 060–065. IEEE, 2011. 2.3, 2.2

[85] Andrea Mannini, Stephen S Intille, Mary Rosenberger, Angelo M Sabatini, and William Haskell. Activity recognition using a single accelerometer placed at the wrist or ankle. *Medicine and Science in Sports and Exercise*, 45(11):2193, 2013. 2.1

[86] Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain adaptation: Learning bounds and algorithms. *arXiv preprint arXiv:0902.3430*, 2009. 2.2.2

[87] Oded Maron and Tomás Lozano-Pérez. A framework for multiple-instance learning. *Advances in Neural Information Processing Systems*, pages 570–576, 1998. 6.1.4

[88] Thomas O Mera, Dustin A Heldman, Alberto J Espay, Megan Payne, and Joseph P Giuffrida. Feasibility of home-based automated Parkinson's disease motor assessment. *Journal of Neuroscience Methods*, 203(1):152–156, 2012. 2.3.1

[89] Michael J. Fox Foundation for Parkinson's Research. Parkinson's disease symptoms, 2013. URL `https://www.michaeljfox.org/understanding-parkinsons/living-with-pd/topic.php?symptoms`. 1

[90] MOSEK ApS. *The MOSEK optimization toolbox for MATLAB manual. Version 9.0.*, 2019. URL `http://docs.mosek.com/9.0/toolbox/index.html`. 4.1.2, 4.1.3

[91] Christiana Ossig, Angelo Antonini, Carsten Buhmann, Joseph Classen, Ilona Csoti, Björn Falkenburger, Michael Schwarz, Jürgen Winkler, and Alexander Storch. Wearable sensor-based objective assessment of motor symptoms in Parkinson's disease. *Journal of Neural Transmission*, 123(1):57–64, 2016. 2.3

[92] Luca Palmerini, Laura Rocchi, Sabato Mellone, Franco Valzania, and Lorenzo Chiari. Feature selection for accelerometer-based posture analysis in Parkinson's disease. *IEEE Transactions on Information Technology in Biomedicine*, 15(3):481–490, 2011. 2.3, 2.2

[93] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010. 2.2

[94] Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210, 2011. 2.2.1, 2.2.3

[95] Parkinson's Disease Foundation. Statistics on parkinson's, 2013. URL `http://www.pdf.org/en/parkinson_statistics`. 1

[96] Shyamal Patel, Konrad Lorincz, Richard Hughes, Nancy Huggins, John Growdon, David Standaert, Metin Akay, Jennifer Dy, Matt Welsh, and Paolo Bonato. Monitoring motor fluctuations in patients with Parkinson's disease using wearable sensors. *IEEE Transactions on Information Technology in Biomedicine*, 13(6):864–873, 2009. 2.3, 2.3, 3.4

[97] Thanneer M Perumal, Meghasyam Tummalacherla, Phil Snyder, Elias Chaibub Neto, E Dorsey, Lara Mangravite, and Larsson Omberg. Remote assessment, in real-world setting, of tremor severity in Parkinson's disease patients using smartphone inertial sensors. In *Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers*, pages 215–218. ACM, 2018. 2.5

[98] Paola Pierleoni, Lorenzo Palma, Alberto Belli, and Luca Pernini. A real-time system to aid clinical classification and quantification of tremor in Parkinson's disease. In *Biomedical and Health Informatics (BHI), 2014 IEEE-EMBS International Conference on*, pages 113–116. IEEE, 2014. 3.4

[99] Benjamin Pittman, Reza Hosseini Ghomi, and Dong Si. Parkinson's disease classification of mpower walking activity participants. In *Engineering in Medicine and Biology Society (EMBC), 2018 Annual International Conference of the IEEE*, pages 4253–4256. IEEE, 2018. 2.5

[100] Sujata D Pradhan, Bambi R Brewer, George E Carvell, Patrick J Sparto, Anthony Delitto, and Yoky Matsuoka. Relation between ability to track force during dual tasking and function in individuals with Parkinson's disease. In *Rehabilitation Robotics, 2009. ICORR 2009. IEEE International Conference on*, pages 885–892. IEEE, 2009. 2.3, 2.3

[101] John Prince and Maarten De Vos. A deep learning framework for the remote detection of Parkinson's disease using smart-phone sensor data. In *Engineering in Medicine and Biology Society (EMBC), 2018 Annual International Conference of the IEEE*, pages 3144–3147. IEEE, 2018. 2.5

[102] John Prince, Siddharth Arora, and Maarten de Vos. Big data in Parkinson's disease: using smartphones to remotely detect longitudinal disease phenotypes. *Physiological Measurement*, 39(4):044005, 2018. 2.5

[103] Blake P Printy, Lindsey M Renken, John P Herrmann, Isac Lee, Bryant Johnson, Emily Knight, Georgeta Varga, and Diane Whitmer. Smartphone application for classification

of motor impairment severity in Parkinson's disease. In *Engineering in Medicine and Biology Society (EMBC), 2014 Annual International Conference of the IEEE*, pages 2686–2689. IEEE, 2014. 2.3, 2.3

[104] C. L. Pulliam, D. A. Heldman, E. B. Brokaw, T. O. Mera, Z. K. Mari, and M. A. Burack. Continuous assessment of levodopa response in Parkinson's Disease using wearable motion sensors. *IEEE Transactions on Biomedical Engineering*, PP(99):1–1, 2017. 2.4

[105] CL Pulliam, SR Eichenseer, CG Goetz, Olga Waln, CB Hunter, J Jankovic, DE Vaillancourt, JP Giuffrida, and DA Heldman. Continuous in-home monitoring of essential tremor. *Parkinsonism & Related Disorders*, 20(1):37–40, 2014. 2.3.2, 3.7

[106] Novi Quadrianto, Alex J Smola, Tiberio S Caetano, and Quoc V Le. Estimating labels from label proportions. *Journal of Machine Learning Research*, 10(Oct):2349–2374, 2009. 6.1.5

[107] Cliff Randell and Henk Muller. Context awareness by analysing accelerometer data. In *Wearable Computers, The Fourth International Symposium on*, pages 175–176. IEEE, 2000. 2.1

[108] Seyed Ali Rokni, Marjan Nourollahi, and Hassan Ghasemzadeh. Personalized human activity recognition using convolutional neural networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 2.2.3

[109] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000. 4.1.4, 2, 4.1.4

[110] Sage Bionetworks. mpower: Mobile Parkinson disease study, 2015. URL `http://parkinsonmpower.org/`. 2.3.1, 2.5

[111] Arash Salarian, Heike Russmann, Christian Wider, Pierre R Burkhard, Franios JG Vingerhoets, and Kamiar Aminian. Quantification of tremor and bradykinesia in Parkinson's disease using a novel ambulatory monitoring system. *IEEE Transactions on Biomedical Engineering*, 54(2):313–322, 2007. 2.4, 2.3.1, 3.4

[112] A Samà, C Perez-Lopez, J Romagosa, D Rodriguez-Martin, A Catala, J Cabestany, DA Perez-Martinez, and A Rodriguez-Molinero. Dyskinesia and motor state detection in Parkinson's disease patients with a single movement sensor. In *Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE*, pages 1194–1197. IEEE, 2012. 2.4

[113] Álvaro Sánchez-Ferro, Morad Elshehabi, Catarina Godinho, Dina Salkovic, Markus A Hobert, Josefa Domingos, Janet MT Van Uem, Joaquim J Ferreira, and Walter Maetzler. New methods for the assessment of Parkinson's disease (2005 to 2015): A systematic review. *Movement Disorders*, 31(9):1283–1292, 2016. 2.3

[114] Enver Sangineto, Gloria Zen, Elisa Ricci, and Nicu Sebe. We are not all equal: Personalizing models for facial expression analysis with transductive parameter transfer. In

Proceedings of the 22nd ACM International Conference on Multimedia, pages 357–366. ACM, 2014. 2.2.2, 4, 4.1.4, 4.1.4, 3, 4.1.4, 4.2.4, 4.3, 5, 8

[115] Patrick Schwab and Walter Karlen. Phonemd: learning to diagnose Parkinson's disease from smartphone data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1118–1125, 2019. 2.5

[116] Ana Lígia Silva de Lima, Tim Hahn, Nienke M de Vries, Eli Cohen, Lauren Bataille, Max A Little, Heribert Baldus, Bastiaan R Bloem, and Marjan J Faber. Large-scale wearable sensor deployment in Parkinson's patients: the Parkinson@home study protocol. *JMIR Research Protocols*, 5(3):e172, 2016. 2.3.2

[117] Ana Lígia Silva de Lima, Tim Hahn, Luc JW Evers, Nienke M de Vries, Eli Cohen, Michal Afek, Lauren Bataille, Margaret Daeschler, Kasper Claes, Babak Boroojerdi, et al. Feasibility of large-scale deployment of multiple wearable sensors in Parkinson's disease. *PLoS One*, 12(12):e0189161, 2017. 2.3.2

[118] Ana Lígia Silva de Lima, Luc JW Evers, Tim Hahn, Nienke M De Vries, Margaret Daeschler, Babak Boroojerdi, Dolors Terricabras, Max A Little, Bastiaan R Bloem, and Marjan J Faber. Impact of motor fluctuations on real-life gait in Parkinson's patients. *Gait & Posture*, 62:388–394, 2018. 2.3.2

[119] Ana Lígia Silva de Lima, Tine Smits, Sirwan KL Darweesh, Giulio Valenti, Mladen Milosevic, Marten Pijl, Heribert Baldus, Nienke M de Vries, Marjan J Meinders, and Bastiaan R Bloem. Home-based monitoring of falls using wearable sensors in Parkinson's disease. *Movement Disorders*, 2019. 2.3.2

[120] André A Spadoto, Rodrigo C Guido, João P Papa, and Alexandre X Falcão. Parkinson's disease identification through optimum-path forest. In *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, pages 6087–6090. IEEE, 2010. 2.3, 2.2

[121] Maja Stikic and Bernt Schiele. Activity recognition from sparsely labeled data using multi-instance learning. In *International Symposium on Location-and Context-Awareness*, pages 156–173. Springer, 2009. 2.3.3

[122] Maja Stikic, Diane Larlus, Sandra Ebert, and Bernt Schiele. Weakly supervised recognition of daily life activities with wearable sensors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(12):2521–2537, 2011. 2.3.3

[123] Masashi Sugiyama, Taiji Suzuki, Shinichi Nakajima, Hisashi Kashima, Paul von Bünau, and Motoaki Kawanabe. Direct importance estimation for covariate shift adaptation. *Annals of the Institute of Statistical Mathematics*, 60(4):699–746, 2008. 2.2.2, 2.2.3

[124] Xu Sun, Hisashi Kashima, and Naonori Ueda. Large-scale personalized human activity recognition using online multitask learning. *IEEE Transactions on Knowledge and Data Engineering*, 25(11):2551–2563, 2012. 2.2.3

[125] Portia E Taylor, Gustavo JM Almeida, Takeo Kanade, and Jessica K Hodgins. Classify-

ing human motion quality for knee osteoarthritis using accelerometers. In *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, pages 339–343. IEEE, 2010. 2.1

[126] Iris Tien, Steven D Glaser, and Michael J Aminoff. Characterization of gait abnormalities in Parkinson's disease using a wireless inertial sensor system. In *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, pages 3353–3356. IEEE, 2010. 2.3, 2.2

[127] Markos G Tsipouras, Alexandros T Tzallas, Dimitrios I Fotiadis, and Spyridon Konitsiotis. On automated assessment of levodopa-induced dyskinesia in Parkinson's disease. In *Engineering in Medicine and Biology Society (EMBC), 2011 Annual International Conference of the IEEE*, pages 2679–2682. IEEE, 2011. 2.3, 2.3

[128] Markos G Tsipouras, Alexandros T Tzallas, George Rigas, Sofia Tsouli, Dimitrios I Fotiadis, and Spiros Konitsiotis. An automated methodology for levodopa-induced dyskinesia: assessment based on gyroscope and accelerometer signals. *Artificial Intelligence in Medicine*, 55(2):127–135, 2012. 2.4

[129] Devis Tuia, Jochem Verrelst, Luis Alonso, Fernando Pérez-Cruz, and Gustavo Camps-Valls. Multioutput support vector regression for remote sensing biophysical parameter estimation. *IEEE Geoscience and Remote Sensing Letters*, 8(4):804–808, 2011. 4.1.4, 4.1.4, 4.1.4, 4.2.4

[130] Eus JW Van Someren, Myrthe D Pticek, Johannes D Speelman, Peter R Schuurman, Rianne Esselink, and Dick F Swaab. New actigraph for long-term tremor recording. *Movement Disorders: Official Journal of the Movement Disorder Society*, 21(8):1136–1143, 2006. 2.3, 2.3

[131] Aner Weiss, Sarvi Sharifi, Meir Plotnik, Jeroen PP van Vugt, Nir Giladi, and Jeffrey M Hausdorff. Toward automated, at-home assessment of mobility among patients with Parkinson disease, using a body-worn accelerometer. *Neurorehabilitation and Neural Repair*, 25(9):810–818, 2011. 2.3, 2.2, 2.3.2

[132] Gary Mitchell Weiss and Jeffrey Lockhart. The impact of personalization on smartphone-based activity recognition. In *Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012. 1, 4, 7.1.1, 7.2

[133] Jason Weston and Chris Watkins. Support vector machines for multi-class pattern recognition. In *ESANN*, volume 99, pages 219–224, 1999. 5.1.2

[134] Timothy J Wroge, Yasin Özkanca, Cenk Demiroglu, Dong Si, David C Atkins, and Reza Hosseini Ghomi. Parkinson's disease diagnosis using machine learning and voice. In *2018 IEEE Signal Processing in Medicine and Biology Symposium (SPMB)*, pages 1–7. IEEE, 2018. 2.5

[135] Jun Yang. MILL: A multiple instance learning library. `http://www.cs.cmu.edu/~juny/MILL`, 2008. Accessed: Sept. 3, 2019. 6.1.4

[136] Ulas Yilmaz. *The Earth Mover's Distance*, 2019. URL `https://www.mathworks.com/`

`matlabcentral/fileexchange/22962-the-earth-mover-s-distance`. 2

[137] Andong Zhan, Max A Little, Denzil A Harris, Solomon O Abiola, E Dorsey, Suchi Saria, and Andreas Terzis. High frequency remote monitoring of Parkinson's disease via smartphone: Platform overview and medication response detection. *arXiv preprint arXiv:1601.00960*, 2016. 2.3.1, 2.3.2

[138] Ada Zhang, Alexander Cebulla, Stanislav Panev, Jessica Hodgins, and Fernando De la Torre. Weakly-supervised learning for Parkinson's disease tremor detection. In *Engineering in Medicine and Biology Society (EMBC), 2017 Annual International Conference of the IEEE*, pages 143–147. IEEE, 2017. 2.3.3, 3.2, 6, 7.1.1

[139] Qi Zhang, Sally A Goldman, et al. Em-dd: An improved multiple-instance learning technique. In *Advances in Neural Information Processing Systems*, volume 1, pages 1073–1080, 2001. 6.1, 6.1.4, 6.1.4

[140] Zhongtang Zhao, Yiqiang Chen, Junfa Liu, Zhiqi Shen, and Mingjie Liu. Cross-people mobile-phone based activity recognition. In *Twenty-second International Joint Conference on Artificial Intelligence*, 2011. 2.2.3

[141] Vincent Wenchen Zheng, Derek Hao Hu, and Qiang Yang. Cross-domain activity recognition. In *Proceedings of the 11th International Conference on Ubiquitous Computing*, pages 61–70. ACM, 2009. 2.2

[142] Chun Zhu and Weihua Sheng. Human daily activity recognition in robot-assisted living using multi-sensor fusion. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 2154–2159. IEEE, 2009. 2.1

[143] Daphne GM Zwartjes, Tjitske Heida, Jeroen PP Van Vugt, Jan AG Geelen, and Peter H Veltink. Ambulatory monitoring of activities and motor symptoms in Parkinson's disease. *IEEE Transactions on Biomedical Engineering*, 57(11):2778–2786, 2010. 2.4