Weakly Supervised Learning Algorithms and an Application to Electromyography

by

Tameem Adel Hesham

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Systems Design Engineering

Waterloo, Ontario, Canada, 2014

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.
I understand that my thesis may be made electronically available to the public.

Abstract

In the standard machine learning framework, training data is assumed to be fully supervised. However, collecting fully labelled data is not always easy. Due to cost, time, effort or other types of constraints, requiring the whole data to be labelled can be difficult in many applications, whereas collecting unlabelled data can be relatively easy. Therefore, paradigms that enable learning from unlabelled and/or partially labelled data have been growing recently in machine learning. The focus of this thesis is to provide algorithms that enable weakly annotating unlabelled parts of data not provided in the standard supervised setting consisting of an instance-label pair for each sample, then learning from weakly as well as strongly labelled data. More specifically, the bulk of the thesis aims at finding solutions for data that come in the form of bags or groups of instances where available information about the labels is at the bag level only. This is the form of the electromyographic (EMG) data, which represent the main application of the thesis. Electromyographic (EMG) data can be used to diagnose muscles as either normal or suffering from a neuromuscular disease. Muscles can be classified into one of three labels; normal, myopathic or neurogenic. Each muscle consists of motor units (MUs). Equivalently, an EMG signal detected from a muscle consists of motor unit potential trains (MUPTs). This data is an example of partially labelled data where instances (MUs) are grouped in bags (muscles) and labels are provided for bags but not for instances.

First, we introduce and investigate a weakly supervised learning paradigm that aims at improving classification performance by using a spectral graph-theoretic approach to weakly annotate unlabelled instances before classification. The spectral graph-theoretic phase of this paradigm groups unlabelled data instances using similarity graph models. Two new similarity graph models are introduced as well in this paradigm. This paradigm improves overall bag accuracy for EMG datasets.

Second, generative modelling approaches for multiple-instance learning (MIL) are presented. We introduce and analyse a variety of model structures and components of these generative models and believe it can serve as a methodological guide to other MIL tasks of similar

form. This approach improves overall bag accuracy, especially for low-dimensional bags-of-instances datasets like EMG datasets.

MIL generative models provide an example of models where probability distributions need to be represented compactly and efficiently, especially when number of variables of a certain model is large. Sum-product networks (SPNs) represent a relatively new class of deep probabilistic models that aims at providing a compact and tractable representation of a probability distribution. SPNs are used to model the joint distribution of instance features in the MIL generative models. An SPN whose structure is learnt by a structure learning algorithm introduced in this thesis leads to improved bag accuracy for higher-dimensional datasets.

Acknowledgements

I would like to thank my supervisors Dr. Ali Ghodsi and Dr. Dan Stashuk. Dr. Ghodsi's knowledge, of which I failed to know the limits, was a great incentive for me to understand how important it is to be aware of not only one specific area of research but also related areas. The insightful discussions I had with him have provided me with guidance and inspiration. Also, Dr. Ghodsi's probabilistic graphical models course was the beginning of an obsession for me.

I am grateful to Dr. Alex Wong. In addition to his important feedback on my thesis, it has been great working with him. His impressive sharpness and profound understanding of both the core and the application side of my thesis, have been invaluable for the progress of this work.

It has been pleasure working with Dr. Dan Lizotte. His broad and deep knowledge as well as his support have kept my enthusiasm alive throughout the thesis.

I would also like to thank the other members of my PhD committee Dr. Marco Valtorta and Dr. Paul Fieguth for their valuable comments and feedback on my thesis.

Finally, thanks to my family without whom this work would not have been accomplished.

Table of Contents

List of Figures

List of Tables

Abbreviations

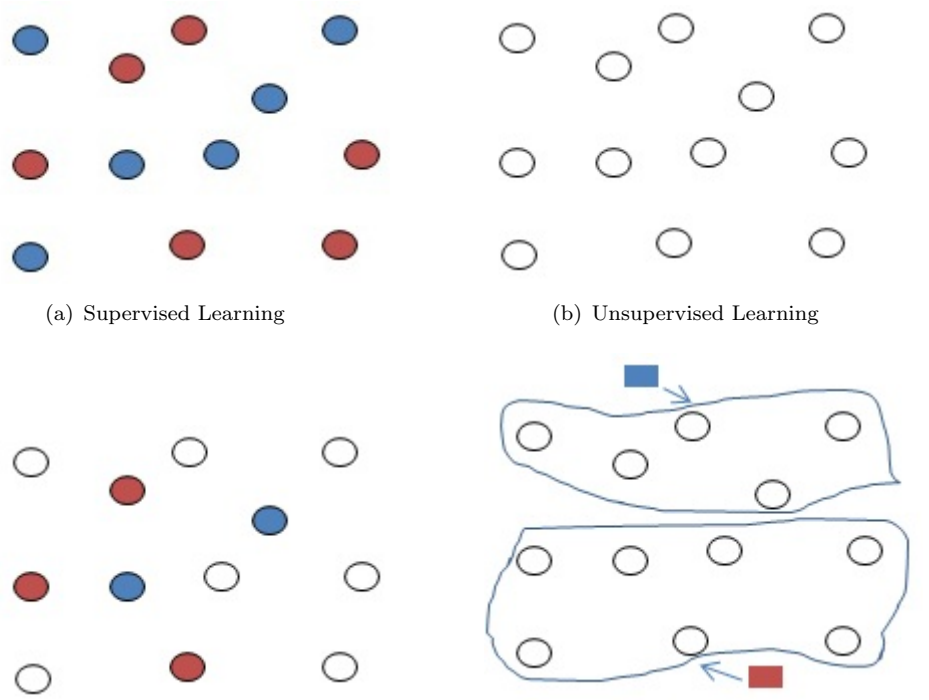
CDSS	**C**linical **D**ecision **S**upport **S**ystem
CPT	**C**onditional **P**robability **T**able
EMG	**E**lectro**M**yo**G**raphy
i.i.d.	**I**ndependent and **I**dentically **D**istributed
IRB	**I**nstitutional **R**eview **B**oard
KDE	**K**ernel **D**ensity **E**stimation
LDA	**L**inear **D**iscriminant **A**nalysis
LOI	**L**evel **O**f **I**nvolvement
MIL	**M**ultiple-**I**nstance **L**earning
MLE	**M**aximum-**L**ikelihood **E**stimation
MU	**M**otor **U**nit
MUP	**M**otor **U**nit **P**otential
MUPT	**M**otor **U**nit **P**otential **T**rain
NMF	**N**onnegative **M**atrix **F**actorization
PDF	**P**robability **D**ensity **F**unction
QDA	**Q**uadratic **D**iscriminant **A**nalysis
R1D	**R**ank-**O**ne **D**owndate
SPN	**S**um-**P**roduct **N**etwork

Chapter 1

Introduction

Two of the most eminent and contrasting statistical machine learning paradigms are supervised and unsupervised learning. In supervised learning (sometimes referred to as learning with a teacher), a learner is given data samples/instances where each instance has its own label and the task of the learner is to implement training on these labelled instances so that previously unseen or test instances can be reliably labelled by the developed learning model [Hastie et al., 2009]. In unsupervised learning, a learner is given unlabelled data instances and the task is to derive a descriptive structure of the given unlabelled data [Hastie et al., 2009]. Nowadays, learning has numerous applications, like spam detection, speech recognition, text categorisation, handwritten digit recognition and many others.

There are other learning paradigms where learning is not fully supervised or, equivalently, where a learner is given non-perfect data for training. The goal of these paradigms is to learn a predictive model based on the available data. Then, test data are to be labelled using this model. Non-perfect (partially labelled) data can take different settings, each having its corresponding learning paradigm(s). The most common paradigm that learns from partially labelled data is semi-supervised learning, which aims at developing models by learning from partially labelled data consisting of labelled as well as unlabelled instances [Zhou and Xu, 2007]. In many problems, using unlabelled or weakly labelled instances in conjunction with labelled instances can lead to an improvement in the performance of the learning model compared to a model based on labelled instances only, and this is the premise on which



(a) Supervised Learning

(b) Unsupervised Learning

(c) Semi-supervised Learning

(d) Bags-of-Instances. Each bag has a label

FIGURE 1.1: Data settings for four Learning Paradigms. **(a)** Supervised Learning: all instances are labelled. **(b)** Unsupervised Learning: all instances are unlabelled. **(c)** Semi-supervised Learning: some instances are labelled and some are unlabelled. **(d)** Bags-of-Instances Learning: Only bags are labelled.

semi-supervised learning is based. The number of unlabelled instances is usually greater than the number of labelled instances in paradigms of this kind. This is a consequence of the relative ease by which unlabelled data can be obtained in numerous applications. Semi-supervised learning is commonly considered a subset paradigm of weakly supervised learning [Joulin and Bach, 2012, Li et al., 2013, Guo et al., 2011] but due to its increasing importance, it is also considered by some (like Guillaumin et al. [2010]) as a related but separate paradigm. Weakly supervised learning generalises other learning paradigms, like paradigms acting on settings where data is given in the form of bags or groups of instances and label information is given at the bag level only. Figure 1.1 depicts the difference between supervised, unsupervised, semi-supervised and bags-of-instances weakly supervised learning. The focus of this thesis is on paradigms where learning is applied on bags of instances. Label information in this case is provided at the bag level only and the introduced algorithms aim

at, first weakly annotating data at the instance level, then applying weakly supervised learning on the resulting weakly labelled data. We are mainly concerned with the bags-of-instances setting but depending on each introduced solution, a subproblem can be cast to another setting and in order to demonstrate the validity of the solution to the subproblem, we may check another data setting that corresponds to the subproblem.

A common example of the bags-of-instances setting is an object recognition image database with binary labels where a positive bag label indicates the existence of a specific object in the corresponding image whereas a negative bag label means that the object does not exist [Blaschko et al., 2010, Bergamo and Torresani, 2010]. A negative labelled image in this case is fully labelled. A positive labelled image can be fully labelled if the exact location(s) of the object is identified, but more often than not this is not the case and positive labelled images are partially labelled each with a latent variable referring to the location of the object in the image.

The main application of the research in this thesis is a muscle classification problem based on electromyographic (EMG) data. EMG muscle data represent partially labelled data where labels are available only at the bag level. Electromyographic (EMG) signals provide a unique insight into both the structure and function of muscles [Farkas et al., 2010] and can therefore be used in a muscle classification problem whose practical aim is muscle diagnosis. An EMG signal of a muscle consists of motor unit potential trains (MUPTs) whereas each MUPT represents the activity of a motor unit (MU) during a muscle contraction. Motor units (MUs) are components of a muscle. Motor unit (MU) activation changes caused by a disorder are reflected in the characteristics of EMG signals allowing them to be used to help diagnose neuromuscular disorders. As shown in the right hand side of Figure 1.2, EMG signals representing the electrical activities of MUs have different characteristics depending on whether the MU is normal, myopathic or neurogenic. More details about the morphological changes in muscles due to neuromuscular disorders are provided in Chapter 2. In EMG datasets, MU labels are based on the label of the muscle from which they were detected and do not consider the actual clinical state of the MU. Therefore, EMG datasets are partially labelled data where labels are provided for the bags but not for individual instances.

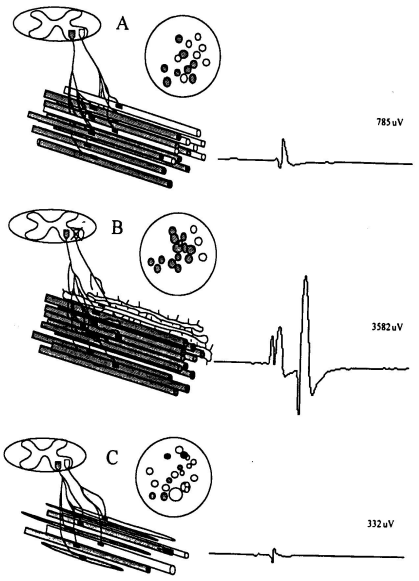


FIGURE 1.2: Schematic representation of the effects of the various categories of neuromuscular disorders [Farkas et al., 2010]. A Normal motor unit (MU), B Neurogenic (MU) & C Myopathic (MU). In the right hand side, motor unit potentials (MUPs) of each label are displayed.

The primary goal of the research in this thesis is to provide bag (muscle) accuracy for the muscle classification problem as well as bag accuracy for similar datasets that share the same bags-of-instances setting but might differ in the assumptions induced by the problem as well as size and dimension of the data.

The main problem therefore is to assign bag labels to data on the bags-of-instances form. There are subproblems as well. In the weakly supervised learning paradigm, there is a grouping subproblem whose solution is tested on other unlabelled datasets. In the generative multiple-instance learning (MIL) modelling paradigm, representing feature probability distributions efficiently is another subproblem. One subproblem that is related to the EMG data only is to get a measure of confidence in muscle characterisations and another of level of involvement (LOI) of motor unit potentials of a muscle.

Figure 1.3 provides a schematic description of the data and paradigms introduced to learn from them.

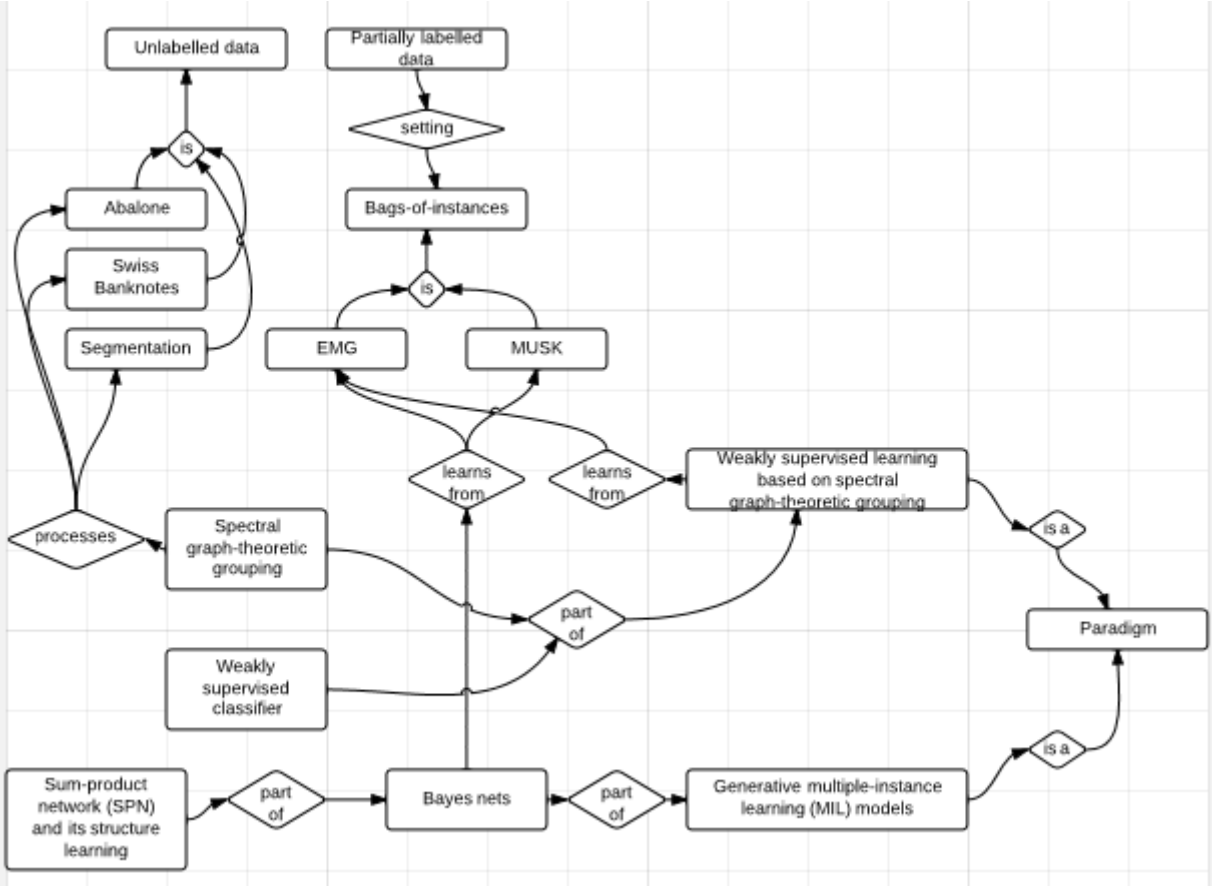


FIGURE 1.3: A representation of data and paradigms in use. EMG datasets are the main datasets in use but there are other datasets, like the MUSK dataset. Both these datasets are bags-of-instances datasets. Other datasets in use, for a subproblem related to one of the paradigms, are Abalone, Swiss Banknotes and Segmentation. The latter three datasets are unlabelled. The first introduced paradigm is a weakly supervised learning paradigm based on spectral graph-theoretic grouping. It is applied on the EMG datasets. It is not applied on the MUSK data because it is not applicable on the first phase of the paradigm which is the spectral graph-theoretic grouping phase. The spectral graph-theoretic grouping phase aims at weakly annotating unlabelled data by first grouping them then assign a label to each group based on the relationship between labelled and unlabelled data. This means that the first step of the spectral graph-theoretic grouping phase is a grouping process. The grouping process is one of the subproblems and this phase on its own is tested on unlabelled datasets; Abalone, Swiss Banknotes and Segmentation. The second phase of the weakly supervised learning paradigm is the weakly supervised classifier used to classify EMG data. The second introduced paradigm is the generative MIL modelling paradigm. In this paradigm, generative models are constructed by Bayes nets. It is applied on both EMG and MUSK datasets. One subproblem of this paradigm is to represent feature probability distributions especially for medium or large dimensional datasets, like the MUSK dataset. Sum Product networks (SPNs) is a model by which probability distributions can be represented efficiently. SPNs are applied as a part of the generative MIL model with the task of modelling the MUSK feature probability distributions.

1.1 Contributions

Each contribution out of the first three contributions represents a strategy or part of a strategy that learns from data of the bags-of-instances setting. We outline these strategies identifying the main characteristics of each.

Weakly Supervised Learning based on Spectral Graph-Theoretic Grouping

In this weakly supervised learning paradigm, a learner receives a training set containing fully labelled bags of instances as well as partially labelled bags of instances. First, a spectral graph-theoretic grouping phase is applied to weakly annotate unlabelled data. Then a weakly supervised classifier is applied on weakly annotated data as well as labelled data. In the first phase, novel similarity graph models are proposed. Results show that the introduced spectral graph-theoretic approach is effective at weakly annotating unlabelled parts of the data, which results in an improved accuracy of the resulting weakly supervised classifier. This weakly supervised paradigm leads to an improved muscle classification accuracy.
The spectral graph-theoretic grouping phase weakly annotates instances of the data, which are instances belonging to bags with known bag labels but unknown instance labels. This leads to the fact it can be used as an unsupervised learning approach because these instances it weakly annotates are unlabelled. Experiments are performed on some benchmark unlabelled datasets and results show that the introduced strategy is effective at constructing a descriptive grouping structure of each of these datasets.

Generative Multiple-Instance Learning (MIL) Models

In its original setting introduced by Dietterich et al. [1997], multiple-instance learning (MIL) referred to a binary class learning paradigm where instances are grouped into bags and a bag is considered positive if it contains at least one positive instance, otherwise a bag is labelled negative. We extend the definition of MIL to cover multi-class problems, introduce an MIL paradigm based on generative models and provide intuition and guidelines for applying these

generative models to MIL problems. A state-of-the-art solution to the problem of muscle classification is provided via the MIL paradigm.

Structure Learning of Sum-Product Networks (SPN)

Features used in the EMG data are 8 features. For bags-of-instances datasets with a much greater number of features, MIL generative models perform relatively well but less well than they do with EMG datasets. Therefore, there is a need for a better way of handling the feature dependence relationships in a more compact and tractable manner than those utilised in the MIL approach introduced above. The most commonly used MIL dataset in the literature is referred to as the MUSK dataset [Dietterich et al., 1997] and it contains 168 features. Therefore, the MUSK dataset represents an example where there is a medium rather than small number of features and where the need to represent dependence relationships among features compactly and efficiently arises. Sum-product networks (SPNs) are probabilistic graphical models that are capable of providing deep architectures that can compactly represent complex dependence relationships among variables while still remaining tractable. A new algorithm for learning the structure of SPNs is introduced and applied on the MUSK dataset. The constructed SPN extends the MIL generative models and bag accuracy results demonstrate that it better expresses dependence relationships.

Regarding the application, the contribution is providing state-of-the-art solutions to the muscle classification problem using both the weakly supervised learning paradigm and generative MIL modelling paradigm on one of the two EMG datasets (upper leg) and using the generative MIL modelling paradigm on the other EMG dataset (lower leg). Also, the bag accuracy of the MUSK dataset that resulted from using SPNs to model instance feature distributions is near state-of-the-art results on the MUSK data.

1.2 Outline

Chapter 2 introduces background material related to the research of the thesis. It contains basic background material about statistical learning as well as unsupervised learning performed on undirected graphs. It also contains background about EMG data. In Chapter 3, the problem is formally defined along with a description of the bags-of-instances partially labelled data of interest. In Chapter 4, the weakly supervised learning paradigm is presented. Chapters 5 and 6 are related to the generative MIL modelling paradigm. Chapter 5 presents the generative MIL models and their results on EMG and MUSK data. Chapter 6 presents the sum-product network (SPN) structure learning algorithm and how it leads to better results on the MUSK dataset when the respective SPN is used to model the instance feature distributions. Finally, concluding comments are given in Chapter 7. Chapters 4, 5 and 6 represent the chapters where the methodology as well as the contributions are. These 3 chapters begin with an overview each then a brief listing of the related work, followed by a description of the introduced algorithm and then the results.

The content of this thesis is mainly based on three publications. The weakly supervised learning paradigm is presented in Adel et al. [2014]. The work on MIL generative models is presented in Adel et al. [2013]. Finally the SPN structure learning algorithm, along with a more comprehensive set of experiments, is presented in Adel and Ghodsi [2014].
Other publications that were developed during the time of the thesis but are secondary to this document, include Adel et al. [2012] and Parsaei et al. [2012].

Chapter 2

Background

Here we present fundamental concepts that are used throughout this thesis. We start with a brief introduction of basic probability rules, then we follow with an overview about statistical learning and finally unsupervised learning performed on undirected graphs. The material in these sections is not comprehensive as it is presented with an eye on its use in the rest of the chapters of the thesis. Finally Section 2.4 presents information about EMG data, which represent the main application of the introduced paradigms.

2.1 Probability and Bayes Rule

A random variable is a variable whose value is not deterministic as it is subject to variability due to randomness [Yates et al., 2003]. A random variable can be assigned a value out of a set of possible values. Confidence that a random variable X has a specific value is denoted by its probability $p(x)$, i.e. $0 \leq p(x) \leq 1$. Note that, as a convention, we denote realisations of random variables in lower case. The function describing the set of values of a random variable and their respective probabilities is referred to as a probability distribution $p(X)$. If the set of possible values X can take is countable, then the probability distribution is a probability mass function (PMF), whereas an uncountable set of values within one or more intervals is represented by a probability density function (PDF). The probability that a random variable Y is equal to a certain value given that another random variable X

is definitely equal to another value, is known as the conditional probability $p(y|x)$. The probability that two random variables X & Y are equal to two respective values is referred to as the joint probability $p(x, y)$. Similar to $p(X)$, $p(Y|X)$ and $p(X, Y)$ can be cast as probability distributions by normalisation. If the probability distributions of $p(X, Y)$ and p(X) are known, then $p(Y|X)$ can be obtained as follows:

$$p(Y|X) = \frac{p(X, Y)}{p(X)} \tag{2.1}$$

As $p(X|Y)$ can be obtained similarly, Bayes rule can now be presented as follows:

$$p(Y|X) = \frac{p(X|Y)\, p(Y)}{p(X)} \tag{2.2}$$

2.2 Statistical Learning

In Chapter 1, we briefly presented a high-level taxonomy of learning paradigms that was mainly based on the setting of the data available for learning. Here we present an example followed by a more formal notion of statistical learning.

Imagine you are given a set of N musical tunes each with its music genre. You are assigned the task of assigning music genres to another set of songs or musical tunes. The latter set of tunes need to be labelled each by one genre. You want to complete the task properly by minimising the number of tunes you ultimately classify incorrectly, out of those in the set. You also want to finish the task as soon as you can, provided that you still do it properly; which means that if you can assign the correct genre to a tune in its first few seconds, this is better than doing the same after two minutes. You decide which tune belongs to which genre, based on the experience you obtained by listening to the N labelled tunes and based on the group of genres from which you pick one for each tune. Your brain chooses features of the tune that are relevant to the genre assigning task at hand. For example, if tunes are to be classified into "classical" or "rock" tunes, your criterion can heavily count on assigning one of the two genres based on whether or not there is singing, not only music. However, if the

group of genres include "folk" as well, you should add tune features that are relevant to the genres, or more precisely, relevant to the difference between a genre and another. Features like time signature and whether or not there are electrical instruments could be used to differentiate "rock" from "folk". Further features should be added in case "alternative rock" (similar to "rock" in all the previously mentioned features) is added to the group of genres; e.g. lyrical style. As you are required to be accurate as well as time efficient, you try to base your genre assignment on features that are relevant to the discrimination among genres as well as non-redundant.

Learning problems of that kind can be represented by a statistical model. We start with describing a statistical learning model.

2.2.1 Statistical Learning Model

Input The learner receives a set of N independent and identically distributed (i.i.d.) instances where each instance X has p features and has a label Y assigned to it. The set of possible values of Y has a cardinality [1] equal to t. This set of instances is referred to as training instances Hastie et al. [2009]. Training data is assumed to be drawn from an unknown distribution.

Sticking to the above example. Input to the learner is a set of tunes, where each tune has p features, e.g. instrumentation, time signature, energy, duration, tempo, etc. Each tune is also assigned a label indicating its genre.

Output The learner is required to build a prediction function that maps X to Y. The prediction function is used to predict an output $\hat{Y}$ of each instance belonging to another set of instances, referred to as the test set. The learner bases the mapping he develops from X to Y on the training instances (supervised learning). In doing so, the learner uses features that he finds relevant to the mapping, i.e. features that provide useful information about the mapping, and non-redundant, i.e. each feature provides further useful information than the already selected features. When t is finite, the prediction function can be referred to as

[1] We assume labels are discrete as regression problems are not discussed in this thesis.

a classifier. Assuming a Bayes classifier, $t = 2$ and $Y \in \{1, 2\}$, if $P(y = 1|x) > P(y = 2|x)$, then $\hat{y} = 1$, and vice versa.

This is the task of assigning genres to each test tune in our example.

A prediction function usually contains parameters β. In case a prediction function fits the training instances too well, there is a risk of overfitting. Overfitting refers to the phenomenon of a classifier that performs accurately on training data but fails to generalise and does not perform well on other data instances. In order to avoid overfitting, β should be chosen so that a classifier has a good balance between its bias and variance. Bias refers to the accuracy of a classifier on the training data instances whereas variance refers to the sensitivity of a classifier to changes in the set of training instances [Geman et al., 1992]. Classifiers that overfit have high variance. The bias-variance tradeoff (also referred to as bias-variance dilemma) has been well studied in machine learning. For example, Hastie et al. [2009], James et al. [2013], Geman et al. [1992] present more comprehensive material on this subject.

Overfitting can be thought of as if you try too hard to adjust your criterion of deciding a genre of a tune based on very specific characteristics of each tune you had in the training set (low bias and high variance) rather than having a better sense of what generally discriminates a genre from another and what is a special characteristic of few training tunes (higher bias and lower variance).

There is more than one measure by which the performance of a classifier can be evaluated. One of them is error rate. Error rate is defined the probability of a misclassification. It is equal to the number of misclassified test instances divided by the total number of test instances. Bayes optimal error rate is defined as the error achieved by an optimal classifier and it represents the theoretical minimum error on a certain dataset [Keinosuke, 1990]. The Bayes optimal error rate is theoretical mainly because the distribution from which the training (and test, assuming they both were drawn from the same distribution) data is drawn, is unknown. Accuracy of a classifier is equal to $1-$ error rate, which is equal to the number of correctly classified test instances divided by the total number of test instances.

Least Squares To build a classifier, its parameters β should be chosen according to a criterion. One criterion aims at minimising the residual sum of squares of errors committed in classifying test instances. This criterion is referred to as least squares. Assuming that $f_\beta(x)$ is the classifier's output for an instance x, least squares can be formulated as follows [Hastie et al., 2009]:

$$minimise\ least\ squares(\beta) = \sum_{i=1}^{N}(y_i - f_\beta(x_i))^2 \tag{2.3}$$

Maximum Likelihood Another criterion, which is based on likelihood, is referred to as maximum likelihood estimation (MLE). Let's first describe the likelihood. Likelihood is a function of the parameters β of a classifier, or more generally of a statistical model. It represents the probability that training instances are equal to their respective values given the parameter values. Likelihood of a classifier can be described as $l(\beta) = p(Y, X|\beta)$. According to maximum likelihood, optimal values of the parameters β are those that make the probability of the observed training samples as large as possible [Hastie et al., 2009]. Because it is more convenient to work with logarithms and they do not change values of β that maximise the likelihood, log-likelihood is usually used rather than likelihood.

Generative Models and Discriminative Models Describing the likelihood function as $l(\beta) = p(Y, X|\beta)$ refers to a generative model where the goal is to learn a joint distribution that underlies the data X as well as the unobserved variable Y that represents their labels. In contrast with generative models, the likelihood function of a discriminative model would be equal to $l(\beta) = p(Y|X, \beta)$ because discriminative models model the conditional probability distribution of an unobserved variable Y given observed data represented by X. One advantage of generative models versus discriminative models is that they can be used in data simulation because they model both the data and the label or unobserved variable. Generative models can also be used in classification because, the joint distribution they provide can be used in classification by applying Bayes rule. On the other hand, discriminative models can sometimes lead to better classification performance than generative

models [Lafferty et al., 2001, Ng and Jordan, 2001], but this depends on the problem and the data. Discriminative models usually perform well when there is plenty of labelled data, but typically do not exploit unlabelled data. In contrast, generative models can make use of unlabelled data in learning via a latent variable model [Bishop and Lasserre, 2007]. Ng and Jordan [2001] reached an important conclusion, related to the asymptotic error of both models when used as classifiers. They noted that while the asymptotic error is lower with discriminative classifiers, generative classifiers reach their asymptotic error much more quickly. A generative classifier can be considered a density estimation problem where it is required to estimate the probability distribution $P(X, Y)$. Similarly, a discriminative classifier can be considered a density estimation problem but the probability distribution to be estimated is $P(Y|X)$ [Hastie et al., 2009]. Examples of generative classifiers are linear discriminant analysis (LDA) and Naive Bayes (NB), while examples of discriminative classifiers include logistic regression and support vector machines (SVM).

Inductive Bias Inductive bias refers to the set of assumptions the learner strictly follows to build the learning function or classifier in our case. Each assumption represents a restriction on the available function space on which the learner builds the learning function, [Ben-David et al., 2011]. One example of an inductive bias is to minimise the number of parameters β in the above generative classifier. This comes as a result of Occam's Razor which favours the simplest consistent rule. One of the benefits of Occam's Razor is that the simplest rule (having minimum number of parameters in our case) can lead to better generalisation, which eventually turns into smaller variance values [Domingos, 1999]. One further advantage of generative models versus discriminative models, as will be explained in more detail in Chapter 5, is that they allow for expert domain knowledge to be incorporated into the model in a straightforward manner, leading to good inductive bias.

2.3 Unsupervised Learning on Undirected Graphs

Back to the example of musical tunes defined in Section 2.2. Imagine you are not given any labelled tunes to prepare a criterion. You are given a set of tunes without genres assigned

to them and are required to provide some summarising or descriptive information of the set. As a starting point, your brain will start looking for major similarities and dissimilarities among the tunes so that you can start providing the required information.
More formally, in unsupervised learning, the learner is given a set X consisting of N instances $x_1, x_2, ..., x_N$ without labels. The task of the learner is to process the data so that a specific form of descriptive statistics about it is obtained. One common form of such statistics is clustering. We focus here on clustering as it is the single approach of unsupervised learning that is targeted in this thesis. Therefore let's describe clustering as follows:

Clustering Input The learner receives a set X of N i.i.d. instances where each instance has p features. Even if it is not always the case, let's assume another number k is given, representing the number of clusters. This is inline with the research in this thesis.

Clustering Output The learner is required to return a partition of the N instances into k disjoint subsets $C_1, C_2, ..., C_k$, where $\bigcup_{i=1}^{k} C_i = X$ [Ben-David et al., 2006]. A good partitioning should minimise pairwise distances among instances of the same subset and maximise pairwise distances among instances of different subsets, so that subsets are homogeneous and well separated, respectively.
Here the focus is on clustering on undirected graphs. We proceed with some related definitions.

Similarity Graphs Assume the similarity between each pair of instances x_i and x_j in X, is indicated by $s_{ij} \geq 0$. Data instances can be represented by a graph $G = (V, E)$ where each data instance x_i is represented by a vertex v_i. The decision whether or not to connect two vertices by an edge, depends on the similarity s_{ij} between the corresponding two instances. If s_{ij} is larger than a certain threshold (indicated by the similarity graph model followed), then the two vertices are connected by an edge whose weight is s_{ij} [von Luxburg, 2007], otherwise the two vertices are not connected. Therefore, for any element w_{ij} of the weighted adjacency matrix W, $w_{ij} \geq 0$.

The degree of a vertex v_i is

$$d_i = \sum_{j=1}^{N} w_{ij} \tag{2.4}$$

The degree matrix D is a diagonal matrix, whose elements are the vertex degrees.
For a subset A, $A \subset V$, size of A can be defined as:

$$|A| = \text{number of vertices in } A \tag{2.5}$$

or

$$vol(A) = \sum_{i \in A} d_i \tag{2.6}$$

The complement of a subset A is denoted by $\bar{A}$.
For any two subsets A and B, $A, B \subset V$, define:

$$W(A, B) = \sum_{i \in A, j \in B} w_{ij} \tag{2.7}$$

In Section 4.2, a description of the most commonly used similarity graph models is provided.

Based on the graph representation of instances, clustering can be formulated as follows: **Graph Clustering** The learner is required to return a partition of the graph into disjoint subsets, or groups of vertices, where edges between vertices of the same group have weights that are as high as possible (homogeneous groups) and edges between vertices of different groups have weights that are as low as possible (well separated groups).

Graph Cut The minimum cut (mincut) of a graph is a partition of the graph whose cut has the smallest possible sum of weights. For a partition consisting of k groups, mincut can be solved by choosing a partition $C_1, C_2, ..., C_k$ such that the following is minimised [Stoer and Wagner, 1997]:

$$cut(C_1, C_2, ..., C_k) = \frac{1}{2} \sum_{i=1}^{k} W(C_i, \bar{C}_i) \tag{2.8}$$

In order to prevent trivial solutions, e.g. one vertex groups, restrictions on the minimum size of a group are imposed. Two common objective functions, each related to one definition of a group size, are listed here. One is referred to as RatioCut [Hagen and Kahng, 1992]. The other objective function is referred to as Ncut [Shi and Malik, 2000].

$$RatioCut(C_1, C_2, ..., C_k) = \sum_{i=1}^{k} \frac{cut(C_i, \bar{C}_i)}{|C_i|} \tag{2.9}$$

$$NCut(C_1, C_2, ..., C_k) = \sum_{i=1}^{k} \frac{cut(C_i, \bar{C}_i)}{vol(C_i)} \tag{2.10}$$

These updated forms turn the mincut problem into an NP hard problem. Spectral clustering can solve relaxed versions of both problems [von Luxburg, 2007].

Before proceeding with a brief description of spectral clustering algorithms, graph Laplacian matrices are introduced first. Recall that D is the degree matrix and W is the weighted adjacency matrix. The unnormalized graph Laplacian is equal to the following:

$$L = D - W \tag{2.11}$$

There are two ways by which a normalized graph Laplacian can be calculated, which are as follows:

$$L_{nor1} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}} \tag{2.12}$$

or

$$L_{nor2} = D^{-1} L \tag{2.13}$$

2.3.1 Spectral Clustering Algorithms

Algorithms of spectral clustering can be mainly divided into three algorithms, as per the graph Laplacian used [von Luxburg, 2007]. For each of them we assume an access to the following as an input.

Input A similarity matrix $S \in R^{N \times N}$ where s_{ij} denotes similarity between x_i and x_j. Number of clusters k is also given.
Output Clusters $C_1, C_2, ..., C_k$ where each instance x_i belongs to one and only one cluster.

Unnormalized Spectral Clustering

- Construct a similarity graph model by one of the algorithms described in Section 4.2 or one of the two introduced similarity graph models described throughout Chapter 4.
- Calculate the unnormalized Laplacian $L = D - W$.
- Calculate the first k eigenvectors $ev_1, ev_2, ..., ev_k$ of L.
- Let $EV \in R^{N \times k}$ be a matrix where columns of EV are $ev_1, ev_2, ..., ev_k$, and let $y_i \in R^k$ be the i^{th} row of EV.
- Cluster y_i, $i = 1, 2, ..., n$ into clusters $C_1, C_2, ..., C_k$ using k-means.

Normalized Spectral Clustering according to Shi and Malik [2000]

- Construct a similarity graph model by one of the algorithms described in Section 4.2 or one of the two introduced similarity graph models described throughout Chapter 4.
- Calculate the unnormalized Laplacian $L_{nor2} = D - W$.
- Calculate the first k generalised eigenvectors $ev_1, ev_2, ..., ev_k$ of the generalised eigenvalue problem $L_{nor2}\ ev = \gamma\ D\ ev$.
- Let $EV \in R^{N \times k}$ be a matrix where columns of EV are $ev_1, ev_2, ..., ev_k$, and let $y_i \in R^k$ be the i^{th} row of EV.
- Cluster y_i, $i = 1, 2, ..., n$ into clusters $C_1, C_2, ..., C_k$ using k-means.

Normalized Spectral Clustering according to Ng et al. [2002]

- Construct a similarity graph model by one of the algorithms described in Section 4.2 or one of the two introduced similarity graph models described throughout Chapter 4.

- Calculate the normalized Laplacian $L_{nor1} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$.
- Calculate the first k eigenvectors $ev_1, ev_2, ..., ev_k$ of L_{nor1}.
- Let $EV \in R^{N \times k}$ be a matrix where columns of EV are $ev_1, ev_2, ..., ev_k$.
- Construct a matrix $T \in R^{N \times k}$, $t_{ij} = ev_{ij} / (\sum_k ev_{ik}^2)^{\frac{1}{2}}$, which represents rows of EV normalized to norm 1. Let $y_i \in R^k$ be the i^{th} row of T.
- Cluster y_i, $i = 1, 2, ..., n$ into clusters $C_1, C_2, ..., C_k$ using k-means.

2.4 EMG Background

In the previous sections, we presented information about the learning techniques that will be used in the algorithms by which the problems are to be solved. Here we give an overview of the EMG data which represent the main application. As the subject is not so common, we give more detailed information in this section than what is required for the introduced paradigms, so that it is easier to understand the whole EMG process.

Human muscles are composed of motor units and each motor (MU) unit is composed of a specific α-motor neuron and the muscle fibres it innervates. A motor neuron innervates the muscle fibres of an MU via the neuromuscular junction (NMJ) formed at the terminal end of each branch of its axon. Voluntary muscle contractions are initiated when the central nervous system recruits MUs by activating their motor neurons, which in turn, via their NMJs, activate their muscle fibres. At each NMJ, a region of transmembrane current is produced across the sarcolemma membrane of its corresponding fibre when the motor neuron is activated (i.e. discharges an action potential). This transmembrane current creates a change in transmembrane potential (or action potential) which propagates along the fibre and initiates/co-ordinates its contraction [Campbell and Reece, 2001]. The currents creating the action potentials of the activated fibres of recruited MUs summate to create dynamic electric fields in the volume conductor in and around muscles. Electrodes placed in these electric fields detect time changing voltage signals which are the electromyographic (EMG) signals discussed in this section. When a muscle is affected by a neuromuscular disorder, characteristics of its action potentials, and as a result of the EMG signals they create, change depending on whether the muscle is affected by a myopathic or neurogenic disorder and the extent to which the muscle is affected. Therefore, quantitative EMG signal analysis can be used to support the diagnosis of neuromuscular disorders. Clinical quantitative electromyography (QEMG) attempts to use the information contained in an EMG signal to classify the muscle from which it was detected in order to support clinical decisions related to the diagnosis, treatment or management of neuromuscular disorders.

2.4.1 Muscle Morphology, Physiology and Electrophysiology

2.4.1.1 Morphological and Physiological Description of a Muscle

MU Structure and Layout

Each muscle consists of muscle fibres. The muscle fibres of a muscle are grouped according to their innervating α-motor neuron. An MU refers to a single α-motor neuron and the muscle fibres it innervates [Adel et al., 2012]. A voluntary muscle contraction is initiated by the activation of motor neurons whose axons propagate action potentials to their terminal ends where they join with a muscle fibre via a NMJ as shown in Figure 2.1. More specifically, a NMJ is the area where the axon terminal of a motor neuron axon innervates a muscle fibre. In a normal muscle, when a motor neuron is activated (i.e. discharges an action potential) each of its innervated muscle fibres are also activated via their respective NMJ. At each NMJ, following the arrival of the action potential at its axon nerve terminal, a region of transmembrane current is produced across the sarcolemma membrane of its corresponding fibre which creates a change in muscle fibre transmembrane potential (or a muscle fibre action potential) which propagates along the fibre and initiates/co-ordinates its contraction. Therefore, in normal muscle, activation of a motor neuron causes all of its innervated muscle fibres to contract and contribute to the force generated by the muscle.

For each muscle, there is a pool (or group) of motor neurons which are activated to produce a voluntary muscle contraction. The number of muscle fibres in a certain motor unit and the diameter of these fibres determine the size of the motor unit or the magnitude of its contribution to the muscle force created. The number of muscle fibres within a motor unit is not constant. Most muscles have large numbers of smaller MUs and smaller numbers of larger MUs.

An MU territory is the cross-sectional area of a muscle in which the fibres of an MU are randomly located. For a normal MU, its MU fibres are randomly positioned throughout its territory. MU territories can be conceived to be circular, with diameters taking values between 10 and 15 mm depending on the size of the MU. In addition, the MU territories of the MUs of a muscle are greatly overlapped. Therefore, in a normal muscle, adjacent muscle

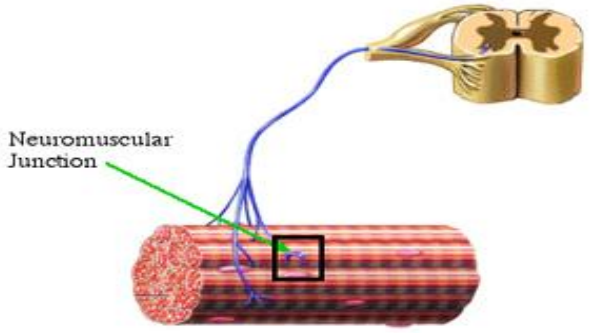


FIGURE 2.1: A motor unit [Robergs and Roberts, 1996].

fibres rarely belong to the same MU. Instead, the muscle fibres of an MU are interdigitated with muscle fibres of many other motor units.

MU Activation

When an MU is recruited, its motor neuron discharges a train of action potentials that propagate along its axon and, as described above, cause the muscle fibres of the MU to contract. The recruitment of only one MU leads to a weak muscle contraction. The recruitment of additional MUs leads to the activation of more muscle fibres and, as a result, muscle contraction becomes gradually stronger.

Motor unit recruitment or derecruitment refers to the activation or deactivation of an MU or population of MUs. Motor unit recruitment strategies vary depending on the inherent properties of the specific motor neuron pools of a muscle. Smaller muscles with smaller pools or numbers of MUs tend to recruit all of their MUs earlier during an increasing force contraction and often have all of their MUs recruited at 30% of maximal voluntary contraction. Larger muscles with large numbers of MUs recruit MUs throughout the entire range of force generation.

2.4.1.2 Muscle Electrophysiology

An innervated muscle fibre is activated when the currents created by the activity of its NMJ create a transmembrane action potential that then propagates in both directions along the muscle fibre away from the NMJ initiating and coordinating contraction of the fibre. In other

words, action potentials propagate along the axon of a motor neuron to activate the muscle fibres of an MU. The currents creating the action potentials of the activated muscle fibres linearly contribute to a spatially and temporally dynamic electric field created in the volume conductor in and around a muscle. The strength and spatial and temporal complexity of the created electric field is determined by the number of MUs active and their size. Electrodes placed in this electric field can be used to detect a time changing voltage signal (i.e. an EMG signal).

2.4.2 EMG Signals

2.4.2.1 Volume Conduction and Detection of EMG Signals

"Volume conduction is the spread of current from a potential source through a conducting medium, such as body tissues" [Dumitru et al., 2002]. Simulation models have been devised so that the effects of having different kinds of volume conductors and arrangements of detection electrodes on an EMG signal can be studied [Malmivuo and Plonsey, 1995].

2.4.3 Neuromuscular Disorders

Neuromuscular disorders change both the morphology and activation patterns of the MUs of the muscles affected. Therefore, the shapes of MUPs detected in muscles affected by neuromuscular disorders will differ from those detected in healthy or normal muscles.
A normal muscle at rest will have no electrophysiological activity (i.e. there will be no electric field created in its surrounding volume conductor). Muscles affected by a neuromuscular disorder can have spontaneous muscle fibre activity called fibrillations and/or spontaneous MU activity called fasiculations.
Myopathic disorders cause muscle fibre atrophy, splitting, hypertrophy and necrosis. Examples of atrophic and hypertrophic muscle fibres are diagrammed in Figure 2.2. Atrophic and split muscle fibres have smaller diameters and slower muscle fibre action potential propagation velocities. Therefore, they typically produce smaller and wider MFPs. Hypertrophic muscle fibres have larger diameters and faster muscle fibre action potential propagation velocities. They therefore produce in general large and narrower MFPs. Necrotic fibres are not

active and do not contribute to detected MUPs. As such, myopathic MUPs, in general, are composed of fewer MFP contributions of varying size and with larger temporal dispersion than in MUPs detected in normal muscles. Myopathic MUPs are therefore generally smaller in size and more complex than normal MUPs. The variation in muscle fibre action potential propagation velocity in a muscle fibre affected by a myopathic process can be greater than normal. This in turn can increase the instability of myopathic MUPs.

Because the MUs of a myopathic muscle are generally smaller during equivalent muscle activations more of them must be recruited and they need to be activated at higher firing rates compared to a normal muscle [Dumitru et al., 2002]. Therefore, at equivalent levels of muscle activation, EMG signals detected in a myopathic muscle can become more complex than EMG signals detected in a normal muscle (see Figure 2.3).

In contrast, neurogenic disorders cause the loss of MUs and muscle fibre denervation. Subsequently, the surviving MUs have increased numbers of fibres with greater spatial fibre densities relative to normal muscle as seen in Figure 2.2. The increased number of MU fibres results in MUPs comprised of larger numbers of MFP contributions. The greater spatial fibre densities result in grouped MFP contributions. Consequently neurogenic MUPs tend to be larger and more complex than normal MUPs sometimes with distinct components or phases (e.g. satellite potentials). During the acute phase of reinnervation newly formed NMJs have larger variations in the time taken to initiate a muscle fibre action potential in their respective muscle fibres. This results in increased instability of neurogenic MUPs.

Because the MUs of a neurogenic muscle are generally larger and because they are fewer in number during equivalent muscle activations, fewer of them must be recruited but they need to be activated at higher firing rates compared to a normal muscle [Dumitru et al., 2002]. Therefore, at equivalent levels of muscle activation, EMG signals detected in a neurogenic muscle can become more complex than EMG signals detected in a normal muscle, as can be seen in Figure 2.3.

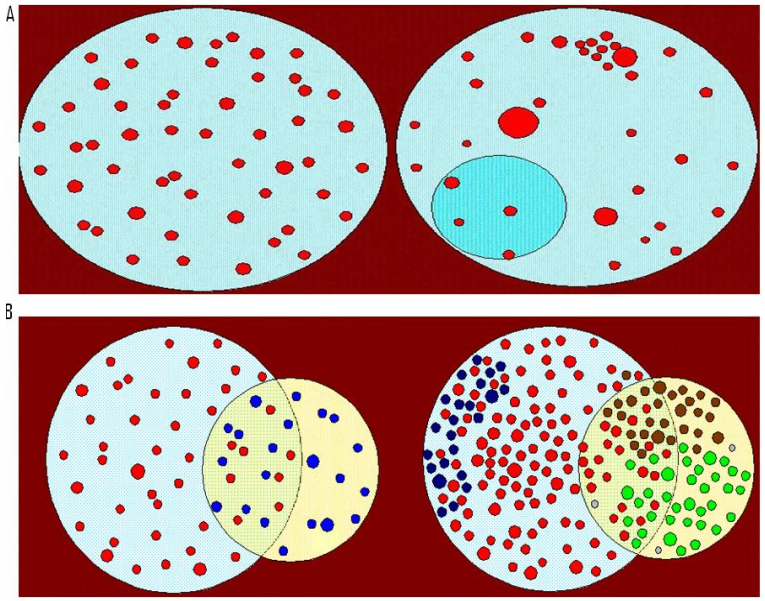


FIGURE 2.2: A. Normal MU vs. myopathic MU.
B. Normal MU vs. neurogenic MU [Barkhaus and Nandedkar, 1994].

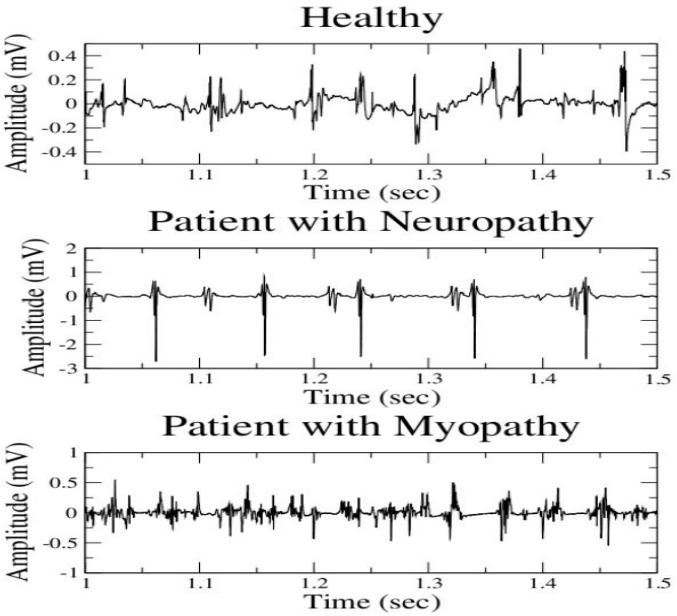


FIGURE 2.3: Examples of healthy, neurogenic and myopathic EMG signals [Zennaro et al., 2003].

2.4.4 How to Extract Clinically Important Information

Suitably detected EMG signals contain information that can be used to assist with the diagnosis of neuromuscular disorders. Specific characteristics of a detected EMG signal can be related to the type of neuromuscular disorder present (i.e. myopathic or neurogenic) as well as the degree to which the muscle may be affected by a disorder. As described above, the changes in MU morphology and activation created by a disease process lead to expected changes in MUP shapes and stability as well as the level of EMG signal complexity. Specific aspects of the detected EMG signals can be analyzed to determine if they were most likely detected in a myopathic, normal or neurogenic muscle. This analysis can be qualitative or quantitative.

2.4.4.1 Qualitative Electromyography

One way of assessing the clinical state of a muscle is to qualitatively analyze EMG signals detected using needle electrodes following abrupt movement of the electrode, while the muscle is at rest and during low levels of muscle activation. Characteristics of the detected signals are subjectively compared to those expected to be detected in normal muscle. The signals detected following abrupt needle movement and while the muscle is at rest are grouped into what is classified as spontaneous activity. Following abrupt needle movement, if the muscle remains active (i.e. significant signals are detected) for a prolonged period of time this is a sign of abnormality. Likewise, if while the muscle is at rest, potentials related to muscle fibre fibrillation or MU fasciculation are detected and the muscle is considered abnormal. MUPs contained in EMG signals detected during low levels of muscle activation are visually analyzed to assess their shape, size and stability.

The objective of this qualitative analysis of the needle-detected EMG signals is to extract information regarding the morphology of a representative sample of MUs of the muscle being examined. Experienced and skilled clinicians can use these qualitative analyses to assist with the diagnosis of an examined muscle with respect to which, if any, specific disease processes may be present and, if present, to what extent.

One of the main disadvantages of qualitative EMG analysis is inter and intra-rater variability. Specific assessments made and the consistency with which they are made depend on the training, experience and skill of the examiner. In addition, no more than a few MUPs can be qualitatively analysed at a time [Farkas et al., 2010]. Therefore, qualitative analysis is restricted to low levels of muscle activation where only a few MUs are recruited and consequently the EMG signals detected are the aggregation of only a few MUPTs.

2.4.4.2 Clinical Quantitative EMG (QEMG)

Quantitative electromyography (QEMG) is an objective assessment of several aspects of detected EMG signals to assist with the diagnosis of a muscle under examination and also to assess the severity of an existing disorder if one is detected. QEMG aims at increasing diagnostic sensitivity and specificity. Unlike qualitative analysis, quantitative analysis is not limited to studying just the first few recruited MUs and EMG signals detected at higher levels of muscle activation can be analyzed. One of the main advantages of quantitative EMG (QEMG) over qualitative EMG is the fact that QEMG techniques attempt to reproducibly represent and interpret features of an EMG signal to extract useful information while qualitative EMG results are not reproducible. QEMG analysis has the advantage of providing greater objectivity and consistency, and is often needed in equivocal cases to increase the certainty of a diagnosis.

Regardless of whether a qualitative or quantitative assessment is used, a similar hierarchical process is followed. First, in order to account for the large variability in motor unit (MU) size and motor unit potential (MUP) shape throughout a muscle, signal features are assessed at several needle positions within a muscle. The data from these various needle positions are then characterized based on whether they possess attributes consistent with certain disease processes. The characterized sampled data are then combined to arrive at an overall impression or characterization of the muscle. Finally, a rule or heuristic is applied to categorize the muscle based on the characterization measures obtained. Training data must be grouped based on a specific muscle group and/or age. Otherwise, the final interpretation of this data may be biased and may have a large margin for error.

To perform QEMG, the complete EMG signal can be analyzed or individual MUP activity can be isolated from an EMG signal using EMG signal decomposition methods. If EMG signal decomposition methods are used, individual MUP can be analyzed which allows information about typical MUP shape, MUP shape stability and MU activation patterns to be used. The next subsection discusses QEMG methods based on individual MUP analysis.

2.4.4.3 MUP Characterization

MUP characterization refers to performing supervised learning to determine if a MUP was created by a normal or abnormal (disordered) MU, if just two labels are considered or by a myopathic, normal or neurogenic MU if three labels are considered. This characterization is based on a training stage that is performed using training data suitably representing each label. MUP features used for MUP characterization often consist of MUP template morphological features; features extracted from the time domain representation of the MUP template as well as spectral features; those extracted from its frequency domain representation [Dumitru et al., 2002]. MU firing pattern features have not yet be effectively used. Typically, a feature selection step is performed to select the best feature subset. As is the case with any supervised learning problem, feature selection can be filter-based (quality metric of the feature subset depends on information content like interclass distance or correlation) or wrapper-based (quality metric of the feature subset depends on the accuracy of the classification process using such feature set). However, wrapper-based feature selection techniques are used more frequently [Pino, 2008].

In addition to the intrinsic MUP template features, like turns, duration, amplitude, etc, combinations of features can be used if they improve the classification results. For instance, MUP template thickness (area/amplitude) can be added to the features used for learning to improve classification performance if it leads to a higher discriminative power of the feature set.

Signal Preprocessing (EMG Signal Decomposition)

EMG signals are the linear summation of the MUP trains created by the MUs active in a muscle. EMG signal decomposition extracts individual MUPTs from an EMG signal. EMG

signal decomposition allows several MUPTs created by MUs concurrently active during a single muscle contraction to be analyzed. The MUPTs extracted during EMG signal decomposition are further analyzed to assist in diagnosing neuromuscular disorders. EMG signal decomposition involves three main steps, described in the following paragraphs.

The first step is to detect the MUPTs comprising an EMG signal. Some EMG signal decomposition algorithms attempt to detect all the MUPTs that existed in the EMG signal while others attempt to extract only MUPTs that had a major contribution to the EMG signal. The following task is to determine the shapes of the different MUPs. This can be done by categorizing the MUPs in the signal based on their shapes and sizes. This categorization, if implemented properly, reveals clusters of MUPs with similar shapes and sizes. As a result, MUPs with different shapes and sizes should belong to different clusters. MUPs with similar shapes and sizes were most probably created by different discharges of the same MU, while MUPs with unique shapes and sizes (i.e. not belonging to a cluster or to a cluster with very few members) are most probably superpositions. The main outcome of this step is to identify the number of MUs that contributed significant MUPs to the EMG signal (i.e. to estimate the number of MUPTs with significant MUPs) and to estimate the MUP template of each discovered MUPT.

The second step is to determine the MU related to each MUP template. Superpositions of MUPs are harder to deal with in the first step as well as in this step. If the overlap is only slight, the constituents might still be recognizable. But if the overlap is complete it might be necessary to try different alignments of the templates to see which gives the closest fit. The motor unit discharge patterns can also be used to help determine which MUs are involved in a superimposed MUP. As discharge rates are assumed to be rather orderly (i.e. IDIs can be assumed to follow a Gaussian distribution), the time at which a particular discharge took place can be estimated from the time at which the preceding or following discharge took place.

The final step in decomposition is to validate the results in order to ensure they are consistent with the expected physiological behaviour of MUs. If there are unexpected short IDI in any of the discharge patterns, or if there are detected MUPs that have not been assigned to a

MUPT, then the decomposition is probably not correct or it is incomplete. On the other hand, if all the activity in the signal (i.e. the detected MUPs) has been adequately accounted for by the set of extracted MUPTs which in turn represent MUs with physiologically realistic discharge patterns, then there is a good chance that the decomposition is substantially complete and accurate.

- **Decomposition Using a Knowledge-Based Certainty Classifier**

  An example of an algorithm that has been developed based on clustering and classification is briefly illustrated here. The algorithm consists of five major steps; signal preprocessing, MUP detection, clustering of a part of the detected MUPs, MUP classification and finally estimation of MUP templates [Parsaei et al., 2012].

  In the beginning, the input EMG signal is filtered using a first-order low pass filter in order to reduce the MUP temporal overlap, accentuate the differences between MUPs created by different MUs and increase the separation between MUPs and the background noise [Parsaei et al., 2012].

  The second major step of the algorithm is MUP detection where the positions of MUPs in the filtered signal are detected using a threshold crossing technique [Stashuk, 1999].

  The third step is based on clustering a part of the data. MUPTs are formed by clustering the the MUPs of this part of the data based on their shapes. The main goal of the clustering step is to estimate the number of MUPTs, estimate their MUP templates and estimate their respective MU firing pattern statistics [Parsaei et al., 2012].

  Afterwards, the rest of the MUPs (those that do not belong to the part of the data used for clustering) are assigned to the extracted MUPTs based on their shapes and consistency between their respective MU firing times and those of the extracted MUPTs. The validity of the extracted MUPTs is assessed and invalid trains are corrected at the end of this step [Parsaei et al., 2012].

  Finally a MUP template is estimated for each MU using mean estimation where a MUP template estimate is calculated by ensemble averaging all the MUPs belonging to its respective MUPT [Stashuk, 1996].

2.4.4.4 Muscle Classification

Aggregation of MUP Characterizations

Characterizations of MUPs must be aggregated to obtain the overall muscle classification of the muscle from which these MUPs were detected. As with MUPs, a set of n muscle likelihood measures is obtained, where n is the number of muscle labels and the muscle is considered to belong to the label that has the highest category likelihood value. In Pfeiffer and Kunze [1995] and Pfeiffer [1999], the idea of implementing probabilistic characterization and aggregating MUP template characterizations using Bayes rule was first introduced. MUP characterization was performed using Fishers LDA. Other techniques used Bayes rule to aggregate MUP template likelihood measures obtained from multiple classifiers like decision trees, LDA and Naive Bayes [Pino, 2008].

Chapter 3

Problem Formulation

3.1 The Bags-of-Instances Setting

Here we describe the bags-of-instances setting of the data upon which the introduced weakly supervised learning algorithms are implemented. Let t be the number of possible bag/instance labels. Let n be the number of bags in the training set. Let $B \in \{1, 2, ..., t\}$ be the random variable representing a bag's label, and let $I_j \in \{1, 2, ..., t\}$ be the random variable representing the label of the jth instance belonging to the bag. The number of instances in the bag, which differs from one bag to another, is denoted by m; the m instance labels are referred to together as the vector $\vec{I}$. Let $\vec{F_j} \in \Re^p$ be the p-dimensional feature vector belonging to the jth instance. Therefore, there is a collection of i.i.d. n bags of the form $(b, \vec{f_1}, \vec{f_2}, ..., \vec{f_{m_\nu}})_\nu, \nu \in \{1, ..., n\}$.

As mentioned earlier, random variables are denoted in upper case while their realisations are denoted in lower case. Elements of a vector are indexed with a square-bracketed subscript, so $\vec{F}_{j[k]}$ is the kth element of the observed feature vector of the jth instance in a bag [Adel et al., 2013]. A "generic" instance label is referred to as I and a "generic" feature vector is referred to as $\vec{F}$.

The main task is to predict the labels of test bags. Feature vectors of instances of test bags are observed. This could be achieved by learning some mapping or concept from the training set in order to correctly label unseen (test) bags. Most weakly supervised learning

and multiple-instance learning (MIL) problems were based on binary class data, like the example shown in Figure 1.1(d). In the muscle classification problem, $t = 3$ for EMG datasets. In other partially labelled datasets of the bags-of-instances setting like the MUSK dataset, $t = 2$. Figure 3.1 displays an example similar to the one shown in Figure 1.1(d) but with three labels, i.e. $t = 3$. The cardinality of bags of each label is usually more than one but they are set to 1 in Figure 3.1 for simplicity.

Again no instance labels are provided. Nonetheless, for all instances I that belong to bags $B = 1$, it is assumed that $I = 1$, while for all instances I that belong to bags $B = 2$ or $B = 3$, value of I is unknown. This assumption is an outcome of the available information about the bags-of-instances datasets of interest in this thesis. For all the EMG muscle datasets, let $B = 1$ denote a normal bag (muscle). It is so rare that a normal muscle would contain myopathic or neurogenic motor units and consequently such motor units can be considered outliers. However, myopathic and neurogenic muscles normally contain normal motor units. Therefore, for $B = 2$ or $B = 3$ (myopathic or neurogenic respectively), I is unknown. The same goes for the MUSK dataset, because negative bags do not contain positive instances. A more detailed description of the MUSK dataset is provided in Chapter 5. Therefore, assume $B = 1$ denotes a negative bag and $B = 2$ denotes a positive bag in the MUSK dataset. This leads to $I = 1$ for all instances belonging to $B = 1$ and unknown I values for all instances belonging to $B = 2$.
In all the introduced learning algorithms, learning is never applied only on the bag level by assuming one representative feature vector for each entire training bag and learning from these vectors. On the contrary, instance labels of a test bag are predicted so that the all important bag label can be predicted.

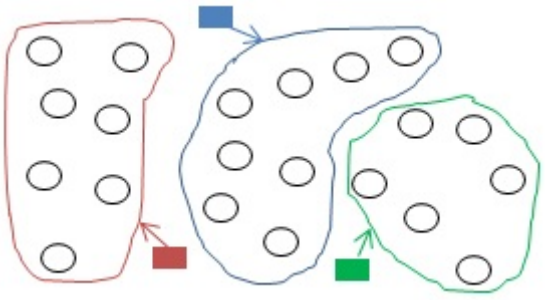


FIGURE 3.1: A 3-class partially labelled dataset where only bags are labelled.

3.2 Characteristics of EMG Muscle Datasets

EMG muscle data used are acquired under IRB approval and sanitized of any personal identifying information and then processed using decomposition-based quantitative electromyography (DQEMG). Data are gathered in a database that is used for both training and testing (using cross-validation). This database contains information from 194 muscles in two groups, as per their location in the body; upper leg and lower leg. There are 99 upper leg muscles and 95 lower leg muscles. Each group is treated as a separate dataset. There are three muscle labels; normal, myopathic or neurogenic. As mentioned before, a motor unit potential train (MUPT) is a train of detected motor unit potentials (MUPs) created by the repeated firing of the same MU. In the database, each MUP is represented by feature values of its MUP template; which is an estimate of its typical or expected MUP shape, as well as stability features that express the stability of the different discharges of this MU within their respective MUPT.

Features of EMG datasets Table 3.1 illustrates the EMG features used in the learning techniques applied on EMG datasets, along with a brief description of each feature.

TABLE 3.1: EMG Feature descriptions.

Feature group	Features	Description
size	duration	the interval from the beginning of the first signal deflection from the baseline to its final return to the baseline of an action potential [Dumitru et al., 2002]
	amplitude	the maximum voltage difference between two points (peak-to-peak) [Dumitru et al., 2002]
	area	area under the curve (measured by μV per ms)
shape	thickness	ratio of area to amplitude
global complexity	length index (RIR)	$\frac{length - 2\,(amplitude)}{2\,(amplitude)}$
	shape width	area / length
	# turns	number of positive and negative peaks
	# phases	discrete number of zero crossings plus one
	fibre count	number of fibres belonging to the MU producing this MUP
local complexity	phase area	$\frac{area}{phases}$
	phase complexity	$\frac{turns}{phases}$
	turn area	$\frac{area}{turns}$
	turn length	$\frac{length}{turns}$
	turn amplitude	$\frac{amplitude}{turns}$
	turn width	$\frac{duration}{turns}$
stability	jiggle	shape variability of a raw MUP recorded with a conventional EMG needle electrode [Dumitru et al., 2002]
	A jiggle	shape variability of a band-pass filtered MUP (2^{nd} derivative of the signal)
	shimmer COV	$\frac{st.\ dev.\ of\ distances\ of\ the\ MUPs\ of\ a\ MUPT\ to\ its\ MUP\ template}{mean\ of\ distances\ the\ MUPs\ of\ a\ MUPT\ to\ its\ MUP\ template}$
composite	size index	$2\ log(amplitude) + \frac{area}{amplitude}$

Chapter 4

Weakly Supervised Learning based on Spectral Graph-Theoretic Grouping

4.1 Motivation

In data of the setting bags-of-instances, bag labels B are provided while instance labels I are not. In EMG datasets, $t = 3$, since a bag or instance can be either *normal* ($B = 1$), *myopathic* ($B = 2$), or *neurogenic* ($B = 3$). All instances belonging to normal bags ($B = 1$) can be labelled normal ($I = 1$) as it is very rare that a normal muscle would contain a disordered (myopathic or neurogenic) component. However, this is not the case with disordered bags as disordered muscles do have some normal as well as disordered components. This means that, in the EMG dataset, instances belonging to normal bags are labelled but instances belonging to myopathic and neurogenic bags are unlabelled. Other bags-of-instances datasets usually have similar setup based on the labelling assumptions resulting from the semantics of the dataset. For example in image datasets indicating the existence of an object, a negative labelled image (meaning the object of interest is not in the image) has all its instances labelled negative. Before the work in this chapter, fully supervised classification was performed on all instances in order to predict labels B of test bags. An instance belonging to a myopathic or a neurogenic bag was assumed to be myopathic or neurogenic respectively. This assumption is not valid because myopathic and neurogenic

bags contain normal instances. Therefore, it is easy to see that performance of any classifier would severely suffer due to this assumption.

In this chapter, a spectral graph-theoretic grouping strategy for weakly supervised classification is introduced, where a limited number of available labelled instances (those belonging to normal bags of the muscle dataset) and a larger set of unlabelled instances (those belonging to myopathic and neurogenic bags) are used to construct a larger annotated training set composed of strongly labelled and weakly labelled instances. Strongly labelled instances are instances that belong to normal bags. Weakly labelled instances are those annotated by the spectral grouping; instances that belong to myopathic and neurogenic bags. Size of the dataset as per number of instances is the same, but size of the dataset as per number of annotated instances becomes larger. A spectral grouping algorithm is applied to create groups within unlabelled instances. Afterwards unlabelled instances get weakly annotated based on the inherent relationship between them and strongly labelled instances. A number of similarity graph models for spectral grouping, including two new similarity graph models introduced in this thesis, are explored to investigate their performance in the context of weakly supervised classification. Experimental results using the EMG muscle data demonstrate that the introduced weakly supervised paradigm can provide significant improvements in classification performance on this data. Also, the introduced spectral grouping algorithm, whose implementation is based on the similarity graphs, is tested on other benchmark datasets. Most of the content of this chapter is presented in Adel et al. [2014].

The main objective of this chapter is to turn a training dataset consisting of a limited number of labelled instances and a larger number of unlabelled instances into a larger annotated set consisting of weakly labelled instances, which are those that were unlabelled, and strongly labelled instances, for the sake of improving classification performance. This is to be achieved via the proposed weakly supervised learning paradigm. Unlabelled data are weakly annotated by applying a spectral graph-theoretic grouping strategy that makes use of strongly labelled instances as well as similarity among instances in order to assign weak labels to the unlabelled instances. Spectral graph-theoretic grouping is based on similarity graph models. In addition to the similarity graph models in the literature, two new similarity

graph models are introduced in this thesis. Weakly labelled and strongly labelled instances form a larger annotated training set. By having larger amounts of annotated training data, and if the spectral grouping process is implemented properly, using a larger annotated training dataset should consequently lead to an improved classification performance compared to the case when an invalid assumption about the unlabelled data is used. We believe this paradigm can serve as a methodological guide to other datasets of the bags-of-instances setting, not only this muscle classification example. However, while the spectral graph-theoretic grouping phase of the introduced paradigm has been tested on other benchmark unlabelled datasets, the weakly supervised paradigm as a whole has been tested on the EMG muscle datasets only.

To put it in a nutshell, the goal is to improve classification performance of bags-of-instances datasets (EMG datasets) by constructing a larger annotated training set, then use it for classification. A weakly supervised paradigm is proposed. The paradigm mainly targets datasets of the bags-of-instances setting, like the one shown in Figure 4.1(A). Each colour denotes a label. Assume green denotes label 1, blue denotes label 2 and red denotes label 3. Instances I belonging to the green labelled bags $B = 1$ (e.g. the bag at top left) are labelled $i = 1$, but instances I belonging to blue and red bags $B = 2$ or $B = 3$ (e.g. the other two) are unlabelled. Instances of all blue labelled bags are grouped together in one similarity graph, i.e. $Graph1$ contains all $I \in B = 2$. The same goes for instances of all red labelled bags, i.e. $Graph2$ contains all $I \in B = 3$. Spectral graph-theoretic grouping is performed by first constructing two similarity graph models, $Graph1$ and $Graph2$ and then performing spectral grouping on each graph (Figure 4.1(B)). As per this step, two similarity graph models are proposed. Using the groups resulting from spectral grouping along with the strongly labelled instances associated within the spectral groups, unlabelled instances are weakly annotated and the result is Figure 4.1(C). The assumption is that total number of green instances in blue (resp. red) bags is less than the number of blue (resp. red) instances. Thus, the group with greater cardinality is assigned the blue (resp. red) label:

For $Graph1$,

$\forall I, I \in Graph1, I \in \{1, 2\}$

$if\ |Grp1| > |Grp2|$

$\forall I, I \in Grp1,\ i = 2$

$\forall I, I \in Grp2,\ i = 1$

$else$

$\forall I, I \in Grp1,\ i = 1$

$\forall I, I \in Grp2,\ i = 2$

For $Graph2$,

$\forall I, I \in Graph2, I \in \{1, 3\}$

$if\ |Grp3| > |Grp4|$

$\forall I, I \in Grp3,\ i = 3$

$\forall I, I \in Grp4,\ i = 1$

$else$

$\forall I, I \in Grp3,\ i = 1$

$\forall I, I \in Grp4,\ i = 3.$

The premise is that by applying an efficient grouping strategy on nodes of each similarity graph, we can weakly (but reliably) annotate the corresponding instances. By doing so, we construct a larger annotated training set. Finally a weakly supervised classifier exploits the whole dataset consisting of strongly labelled data and weakly labelled data (Figure 4.1(D)). The rest of the text in this chapter is dedicated to thoroughly study how to apply an "efficient" grouping strategy so that unlabelled instances can be "reliably" annotated.

4.1.1 Related Work

There are numerous examples of learning algorithms where unlabelled or weakly labelled data are utilised [Arora et al., 2007, Chum and Zisserman, 2007, Lee and Grauman, 2011, Winn and Jojic, 2005, Crandall and Huttenlocher, 2006]. Also, Bergamo and Torresani [2010] provide another example where they exploit weakly annotated web images to build

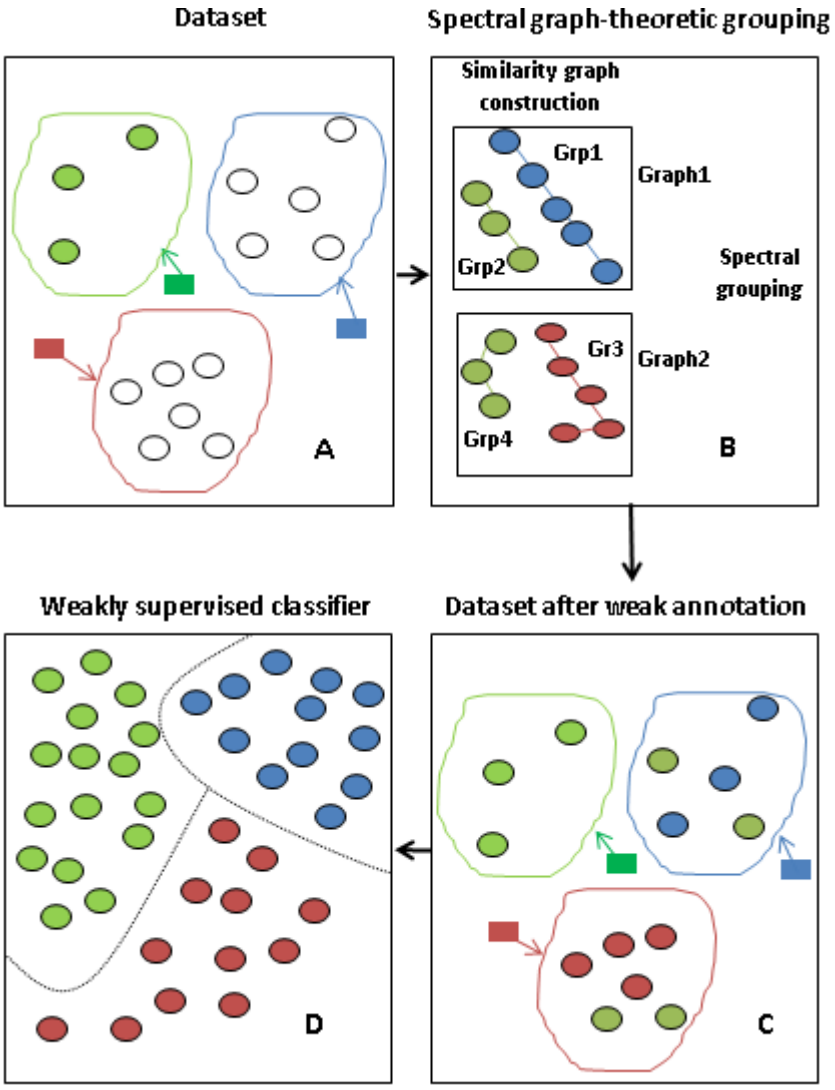


FIGURE 4.1: A schematic representation of the main steps of the proposed weakly supervised paradigm. **A.** A dataset of the bags-of-instances setting (only one bag of each label is shown for simplicity but cardinality of bags of each label is greater). Each bag label is represented by a colour. **B.** All instances of blue bags are grouped in one similarity graph model (Graph1) and the same for instances of red bags (Graph2). Spectral grouping is performed on each similarity graph model to group the instances in two groups. Relation with the green labelled instances decide the label of each group. In this example, the assumption is that total number of green instances in blue (resp. red) bags is less than the number of blue (resp. red) instances. Thus, the group with greater cardinality is assigned the blue (resp. red) label. **C.** Now instances of blue and red bags are weakly annotated while instances of green bags are strongly labelled. **D.** A classifier learns from all instances (weakly and strongly labelled) of all training bags

a weakly supervised object classifier. In addition to object recognition in images, weakly supervised learning has other applications in computer vision. For example, Prest et al. [2012] perform learning based on weakly labelled videos and fully labelled images in order to detect objects from web videos. Other examples of weakly supervised learning algorithms that are applied on videos include Ali et al. [2011] and Leistner et al. [2011].

Graph-theoretic grouping has also been studied before in the literature. Fowlkes et al. [2004] used spectral graph-theoretic grouping in an image segmentation application. They developed spectral groups which were based on using a small number of samples and extrapolating so that the computational requirements are reduced. Aksoy and Haralick [1999] developed a graph-theoretic clustering algorithm that was used for image grouping. They grouped images based on the observation that visually similar images are also similar in the feature space because they have similar feature vectors. Wu and Leahy [1993] represent one further example of a graph-theoretic based clustering algorithm where they develop an algorithm for image segmentation. They performed clustering by building an undirected graph using data instances, then forming mutually exclusive subgraphs by gradually removing arcs according to a certain criterion.

4.2 Methodology

One strategy of weakly annotating unlabelled data is to apply a grouping technique so that each part of the unlabelled data can be related to a group, which in turn is assigned a certain label depending on the inherent relationship between the strongly labelled part of the data and the unlabelled part. Due to the fact that the spectral grouping process acts only on the unlabelled part of the data, the introduced spectral grouping algorithms are applicable on fully unlabelled datasets because they practically act on the unlabelled part of data, as long as other clustering issues (e.g. number of clusters) can be handled via the problem assumptions. This is the case with the datasets used in this weakly supervised paradigm. For the EMG muscle datasets, instances belonging to each disordered type of bag (myopathic or neurogenic) are known to be either normal or of the same disorder. Therefore, there are

two groups/clusters. For the benchmark datasets used, number of clusters is known. As a preliminary phase of the work in this chapter, spectral clustering as well as other clustering algorithms were performed on the EMG muscle datasets. Normalized spectral clustering according to Shi and Malik [2000] performed slightly better than other spectral clustering algorithms which in turn performed better than other clustering algorithms. However, the improvement provided by Shi and Malik's normalized clustering was not considerable. After careful inspection of the reason why Shi and Malik's normalized spectral clustering does not perform better than it does, it turned out to be the fact that all similarity graphs used before in the spectral clustering literature do not capture well the pairwise similarities between instances of the dataset. It is worth noting that there are two main normalized spectral clustering algorithms, one is according to Shi and Malik [2000] and the other is according to Ng et al. [2002]. For the sake of simplicity, the former will be shortly referred to in this chapter as normalized spectral clustering, unless stated otherwise.

4.2.1 Similarity Graph Models

The first step of the proposed weakly supervised paradigm is to perform spectral graph-theoretic grouping on the unlabelled part of the dataset. Spectral graph-theoretic grouping in turn begins by forming similarity graph model(s) of unlabelled data. In the literature, there are several popular similarity graph models that transform a given set $I_1, ..., I_N$ of data instances with pairwise similarities s_{ab}, where a and b are indices, or pairwise distances d_{ab} into a graph. When constructing a similarity graph model the goal is to model the local neighbourhood relationships between the data instances. The following is a list of the main similarity graph models.

The ϵ-neighbourhood graph: Instances that have pairwise distances among each other less than ϵ are connected while the instances with pairwise distances greater than or equal to ϵ are not. Weights are considered to be at the same scale of distances; which is at most ϵ [von Luxburg, 2007]. Therefore ϵ-neighbourhood graph is unweighted.

k-nearest neighbour graphs: An instance I_a is connected to an instance I_b if I_b is among the k-nearest neighbours of I_a. However, the neighbourhood relationship is not symmetric

and due to that the resulting graph is a directed graph. Therefore, the graph should be transformed into an undirected graph. One way of transforming it into an undirected graph is by connecting two instances I_a and I_b if I_b is among the neighbours of I_a "or" I_a is among the neighbours of I_b. The resulting undirected graph is referred to as the k-nearest neighbour graph. Another way is to connect two instances I_a and I_b if I_b is among the neighbours of I_a "and" I_a is among the neighbours of I_b. The resulting undirected graph in this case is referred to as the mutual k-nearest neighbour graph. After building the similarity graph in both cases, edges of the graph are weighted by measuring the similarity of the respective vertices [von Luxburg, 2007].

The fully connected graph: All instances are connected to one another; in other words all instances are considered "similar" to one another. The edges are weighted by s_{ab}. The graph is useful only when local neighbourhoods can be modelled by the similarity function because this is the only way by which the fully connected graph can represent the local neighbourhood relationships [von Luxburg, 2007]. The Gaussian similarity function $s(I_a, I_b) = exp(-||I_a - I_b||^2/(2\sigma^2))$ is an example of this kind of similarity function. The parameter σ of the Gaussian similarity function controls the width of the neighbourhoods. The parameter σ acts like the parameter ϵ in the construction of the ϵ-neighbourhood graph [von Luxburg, 2007].

EMG datasets provide examples of datasets where neither the ϵ-neighbourhood graph, k-nearest neighbour graphs nor fully connected graphs can capture properly the similarities between the data instances mainly because there are different densities within the same dataset. The proposed similarity graph models aim at being robust in handling different data densities within a dataset. They are referred to as probabilistic threshold similarity graph and probabilistic criteria similarity graph. One of the main advantages of the proposed probabilistic threshold and probabilistic criteria similarity graphs is that they do not have a problem in dealing with instances in different scales. This means that unlike the ϵ-neighbourhood graph which does not connect instances belonging to the same scale when a dataset is on different scales, and unlike the k-nearest neighbour graph which, in the same

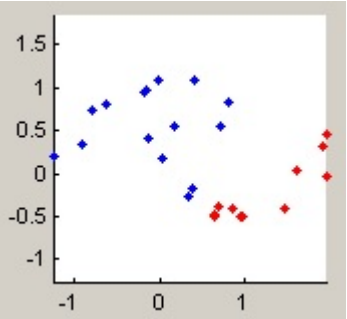
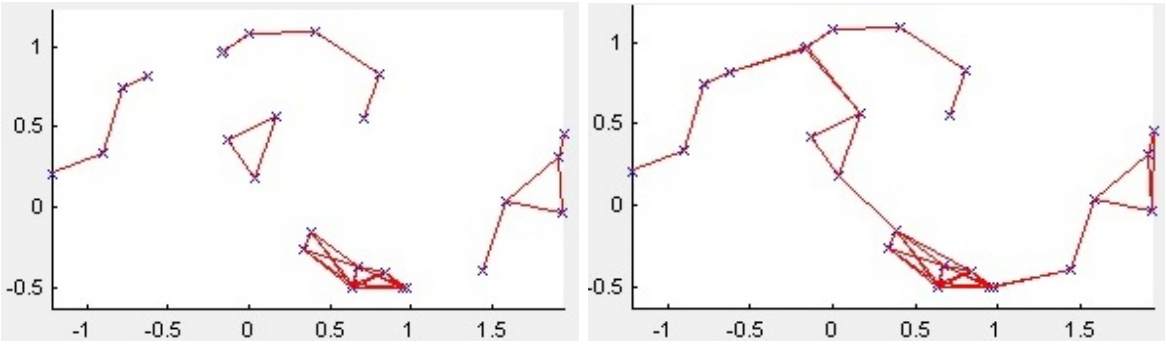


FIGURE 4.2: *DatasetA*: A 26-instance 2-label dataset used to test different similarity graph models. Instances are coloured according to their labels.

case, would connect instances on different scales, the probabilistic threshold similarity graph can connect instances within regions of constant density when data is on different scales. Mutual k-nearest neighbour graph can at times act on different scales but setting the parameter k in this case usually is a problem because, first finding the optimal k value for a certain dataset is tricky and second and more importantly, one dataset can have the optimal value of k that does not mix the data scales with one another on one part of the dataset different from the optimal value of k on another part of the same dataset.

Examples where the advantages of the probabilistic threshold and probabilistic criteria similarity graphs are clear, usually relate to clusters that have irregular shapes. For example Figure 4.2 shows a toy dataset representing a pattern that takes place quite often in the EMG datasets as well as other datasets where there are two or more irregular clusters in the data. A Matlab GUI, which was developed by Hein and Luxburg in [Hein and von Luxburg, 2007], is tailored in order to show the figures used throughout this illustrative example.

Figure 4.3 shows how the instances are connected when an ϵ-neighbourhood graph is used with values of ϵ equal to 0.2298 and 0.2791. When ϵ is less than the former, number of connected components is ≥ 5 whereas number of connected components is always 1 for values of ϵ greater than the latter. The former is the value of ϵ that resulted from leave-one-out cross-validation on this small dataset and it led to 5 connected components as it loosely or never connects instances belonging to the same cluster while bigger values of ϵ overconnected instances belonging to different connected components. Figure 4.4 shows the similarity graphs when a symmetric k-nearest neighbour graph is used with values of k



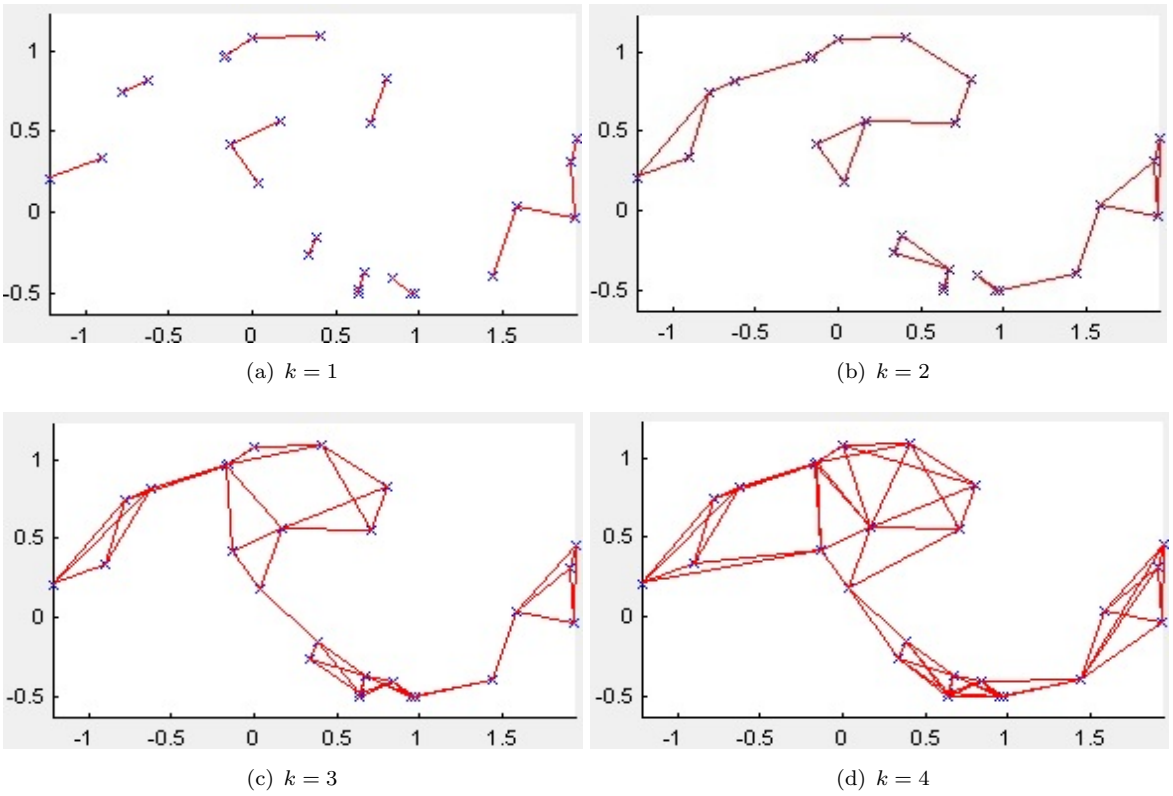
(a) $\epsilon = 0.2298$ (b) $\epsilon = 0.2791$

FIGURE 4.3: Similarity graph of *DatasetA* as a result of applying an ϵ-neighbourhood graph. This technique fails to identify the 2 groups of *DatasetA*.

equal to 1, 2, 3 and 4. Number of connected components is always 1 for values of k greater than 4. No value of k made symmetric k-nearest neighbour graphs get the correct groups or connected components. The value of k resulting from leave-one-out cross-validation is $k = 3$ and it connects instances belonging to different connected components.

Figure 4.5 shows the similarity graphs when a mutual k-nearest neighbour graph is used with values of k equal to 4, 5, 6 and 7. Number of connected components is always 1 for values of k greater than 7. No value of k made mutual k-nearest neighbour graphs get the correct connected components. The value of k resulting from leave-one-out cross-validation is $k = 4$ and even if it is a better fit than both the ϵ-neighbourhood graph and the symmetric k-nearest neighbour graph, the two disconnected components on the right side of Figure 4.5 (a) should have been connected as they belong to the same cluster and the same goes for the component on the right side along with the one in the middle of Figure 4.5 (a).

Figure 4.6 shows the similarity graph resulting from the probabilistic thresholding algorithm with a value of w equal to 0.073 which is the value resulting from applying leave-one-out cross-validation on this illustrative dataset. The probabilistic threshold similarity graph is the only similarity graph model that leads to the correct connected components because the values that w are compared to are normalized values representing the distance between a certain instance and another divided by summation of distances between the former and all instances of the dataset. This normalization leads to similarity graph model that not



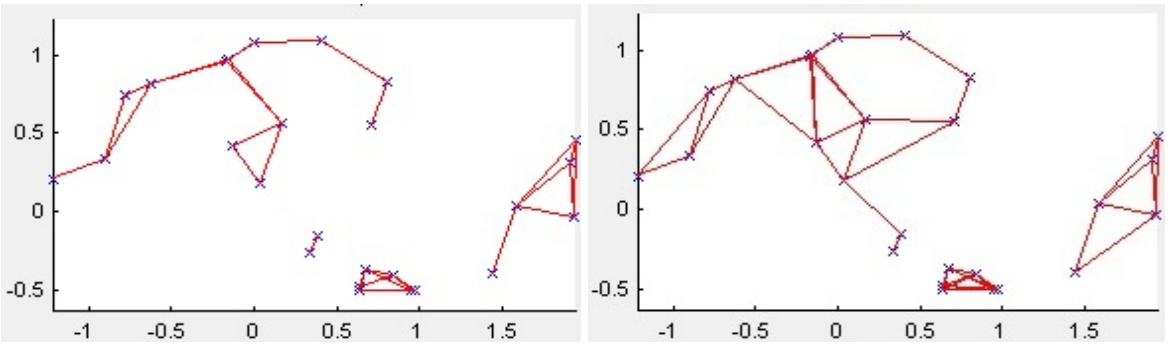
(a) $k = 1$ (b) $k = 2$

(c) $k = 3$ (d) $k = 4$

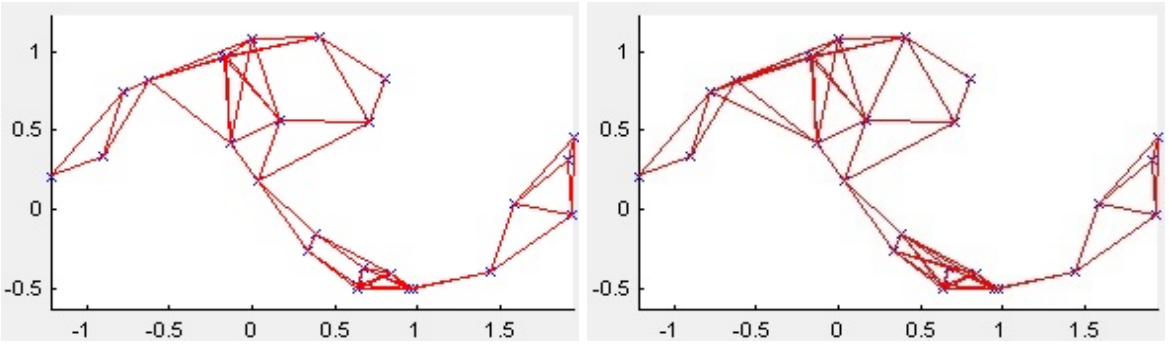
FIGURE 4.4: Similarity graph of *DatasetA* as a result of applying a symmetric k-nearest neighbour graph. This technique fails to identify the 2 groups of *DatasetA*.

only depends on absolute values of parameters but is also heavily impacted by relative weight values where a certain distance value from a certain instance to another is taken into consideration only with relative to another distance from the former instance to a third instance.

In the probabilistic threshold similarity graph, a parameter w is used as a threshold on the similarity values. Similarity values greater than or equal to w are kept while similarity values smaller than w are assigned using a truncated Gaussian distribution with mean $= w$ and standard deviation $= \sigma$. Another parameter ϵ is used to decide the final similarity values as illustrated in Section 4.2.1.1. As shown in equations 4.1 and 4.2, initial values of similarity, which are compared with w are normalized based on the summation of distances from a certain instance. This leads to the fact that the thresholding applied here is relative to the data and does not depend on absolute values as is the case with ϵ in ϵ-neighbourhood graph and



(a) $k = 4$ (b) $k = 5$



(c) $k = 6$ (d) $k = 7$

FIGURE 4.5: Similarity graph of *DatasetA* as a result of applying a mutual k-nearest neighbour graph. This technique fails to identify the 2 groups of *DatasetA*.

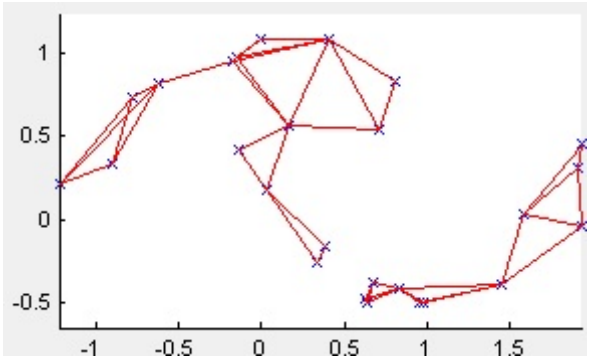


FIGURE 4.6: Similarity graph of *DatasetA* as a result of applying a probabilistic threshold graph with $w = 0.073$. This technique manages to correctly identify the 2 groups of *DatasetA*.

k in the k-nearest neighbour graphs. Nonetheless w and ϵ still provide a hard thresholding on the similarity values and therefore these parameters can still affect the similarity values a great deal. In order to alleviate this effect of hard thresholding, a similarity graph that is based on a probabilistic acceptance criterion is proposed. As illustrated in Section 4.2.1.2, similarity values are assigned either from $s_{ab} \sim N(w, \sigma)$ or to 0 based on a probability value and there is no hard threshold under which similarity values are directly assigned a value of 0.

The proposed approach for constructing similarity graphs can be formally described as follows:

4.2.1.1 Probabilistic Thresholding

Each vertex v_a of the similarity graph model represents an instance I_a. For each vertex v_a, distances between a vertex v_a and all other vertices; $d_{ab}, b = 1, ..., N, b \neq a$, are calculated first as Euclidean distances. Initial similarity values are subsequently calculated as a function of the distances by equation 4.1

$$\mathrm{s}_{ab}^{init} = \frac{{d_{ab}}^{1-m}}{\sum_{b=1}^{N} {d_{ab}}^{1-m}}, \; m > 1 \; is \; a \; smoothing \; parameter \tag{4.1}$$

Similarity values which are greater than or equal to w are kept while the rest of similarity values are assigned using a truncated Gaussian distribution with mean = w and standard deviation = σ. This means that for the interval $s_{ab} \in (0, w)$, values of $s_{ab} \sim N(w, \sigma)$ are used to decide the final value of s_{ab} as follows. If a weight value generated by $N(w, \sigma)$ is smaller than a certain small threshold value ϵ, then the respective similarity value is set to 0, otherwise the similarity value is set to the generated weight. In summary, let's assume that the value s_ϵ leads to a probability density function (PDF) value of ϵ. Then, similarity values greater than or equal to w are taken as they are and similarity values smaller than s_ϵ are set to 0, while for interval $s_{ab} \in (\mathrm{s}_\epsilon, w)$ similarity values are obtained by $s_{ab} \sim N(w, \sigma)$, as shown in equation 4.2. Similarity graphs constructed by the probabilistic thresholding

algorithm are referred to as probabilistic threshold similarity graphs.

$$s_{ab} = \begin{cases} \mathrm{s}_{ab}^{init} & \text{if } \mathrm{s}_{ab}^{init} \geq w \\ f(\mathrm{s}_{ab}^{init}, w, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \mathrm{e}^{-\frac{(\mathrm{s}_{ab}^{init} - w)^2}{2\sigma^2}} & \text{if } \epsilon \leq f(\mathrm{s}_{ab}^{init}, w, \sigma) < w \\ 0 & \text{if } f(\mathrm{s}_{ab}^{init}, w, \sigma) < \epsilon. \end{cases} \tag{4.2}$$

4.2.1.2 Probabilistic Acceptance Criterion

Here, distances and corresponding initial similarity values are calculated the same way as in thresholding. Similarity values that are greater than or equal to w are again kept as they are while a truncated Gaussian distribution with mean $= w$ and standard deviation $= \sigma$ is utilised as follows in order to calculate similarity values smaller than w. The weight values resulting from $N(w, \sigma)$ are accepted as they are into the neighbourhood with a probability based on the generated weight and therefore a stochastic acceptance criterion, that does not require a threshold, is provided. To sum it up, similarity values greater than or equal to w are taken as they are while for interval $s_{ab} \in (0, w)$ similarity values are assigned either by $s_{ab} \sim N(w, \sigma)$, with a probability based on the weight generated from $N(w, \sigma)$, or set to 0 otherwise, as displayed in equation 4.3. Similarity graphs constructed by the probabilistic acceptance criterion algorithm are referred to as probabilistic criterion similarity graphs.

$$s_{ab} = \begin{cases} \mathrm{s}_{ab}^{init} & \text{if } \mathrm{s}_{ab}^{init} \geq w \\ f(\mathrm{s}_{ab}^{init}, w, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \mathrm{e}^{-\frac{(\mathrm{s}_{ab}^{init} - w)^2}{2\sigma^2}} & \text{with prob. } \propto \frac{1}{\sigma\sqrt{2\pi}} \mathrm{e}^{-\frac{(\mathrm{s}_{ab}^{init} - w)^2}{2\sigma^2}} \\ 0 & \text{with prob. } \propto 1 - \frac{1}{\sigma\sqrt{2\pi}} \mathrm{e}^{-\frac{(\mathrm{s}_{ab}^{init} - w)^2}{2\sigma^2}}. \end{cases} \tag{4.3}$$

Both neighbourhood relationships of the introduced similarity graphs are turned into symmetric neighbourhoods in a fashion similar to the k-nearest neighbour graph; either by assigning the maximum value out of $similarity(v_a, v_b)$ & $similarity(v_b, v_a)$ to both of them

or by taking the minimum value out of these two values to be their updated symmetric similarity value.

4.2.2 Spectral Grouping

The graph clustering notation presented in Section 2.3 is repeated here, as well as a listing of the main graph Laplacian matrices in the literature.

Clustering Input The learner receives a set X of N i.i.d. instances where each instance has p features. Even if it is not always the case, but let's assume another number k is given, representing the number of clusters. This is inline with the work in this chapter.

Clustering Output The learner is required to return a partition of the N instances into k disjoint subsets $C_1, C_2, ..., C_k$, where $\bigcup_{i=1}^{k} C_i = X$ [Ben-David et al., 2006]. A good partitioning should minimise pairwise distances among instances of the same subset and maximise pairwise distances among instances of different subsets, so that subsets are homogeneous and well separated, respectively.
As spectral clustering is what is of interest here. Clustering with undirected graphs where vertices represent instances is the form of clustering in use in the work in this chapter.
Graph Clustering The learner is required to return a partition of a graph into disjoint subsets, or groups of vertices, where edges between vertices of different groups have weights that are as low as possible (well separated groups) and edges between vertices within the same group have weights that are as high as possible (homogeneous groups).
Let D be the degree matrix and W be the edge weight matrix of the similarity graph. The unnormalized graph Laplacian is equal to the following:

$$L = D - W$$

There are two ways by which a normalized graph Laplacian can be calculated, which are as follows:

$$L_{nor1} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$$

or

$$L_{nor2} = D^{-1}L$$

A normalized graph Laplacian is calculated in all the algorithms performed in the work in this chapter via the latter equation $L_{nor2} = D^{-1}L$ [Chung, 1997].

4.3 Experiments

There is no ground truth labelling available for the unlabelled instances of the EMG datasets. The main purpose of weakly annotating these unlabelled instances is to improve the performance of the subsequent weak classifier. This means that the accuracy of the weak classifier is the main metric for measuring the quality of the weak annotation. Still, we show another metric which is a clustering internal evaluation measure in order to demonstrate the quality of the grouping process in a generic sense. Davies-Bouldin index is used as an internal evaluation measure for the EMG datasets. Benchmark clustering datasets used in the experiments have their ground truth labels available. Ground truth labels were never used in the learning process by any means. F-measure is used as an external evaluation measure for these datasets where ground truth labels are available.

For 2-cluster problems like the EMG clusterings, Davies-Bouldin can be calculated as follow:

$$\text{Davies-Bouldin} = \frac{1}{2}\sum_{a=1}^{2} max_{a\neq b}\left(\frac{\sigma_a + \sigma_b}{d(c_a, c_b)}\right)$$

where c_a is the centroid of cluster a, σ_a is the average distance of all instances of cluster a to centroid c_a and $d(c_a, c_b)$ is the distance between two centroids [Davies and Bouldin, 1979]. As the numerator expresses the compactness of the clusters of a clustering result (intra-cluster distance) and the denominator expresses the separation among the clusters of the clustering (inter-cluster distance), the smaller the value of Davies-Bouldin index, the better the corresponding clustering.

F-measure is a clustering external evaluation measure that weights the recall by a parameter β [Rijsbergen, 1979]. Precision is $P = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$ and recall is $R =$

$\frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$. F-measure is calculated by:

$$\text{F-measure} = \frac{(\beta^2 + 1)\ P\ R}{\beta^2 P + R}$$

Here we use $\beta = 1$. Therefore, F-measure used here is the harmonic mean of precision and recall:

$$\text{F-measure} = \frac{2\ P\ R}{P + R}$$

Best value of F-measure is 1 or 100% and worst value is 0. In this range, the larger F-measure, the better the corresponding clustering result.

Results are divided into spectral graph-theoretic results, which are based on similarity graph models, and weakly supervised classification results. The former displays the outcome of Section 4.3.1 which provides an analysis of similarity graph models, two of which are introduced in this work. The second part of the results compares between weakly supervised classifiers and the corresponding fully supervised classifier (Section 4.3.2).

4.3.1 Analysis of Similarity Graph Models

Regarding the EMG muscle datasets, the parts used of every muscle dataset in the similarity graphs represent the unlabelled parts. Outcome of each spectral graph-theoretic grouping consists of two groups. The labelled part of every muscle dataset is used to annotate the groups because the group with a smaller number of elements is assigned the label normal while the group with more elements is given the disordered label. This assumption is based on the structure of a disordered muscle, which typically contains more disordered motor units (MUs) than normal MUs. The elements which get annotated by the spectral grouping represent weakly labelled data that can be used as labelled data for the weakly supervised classification algorithm illustrated in the Section 4.3.2. Regarding the features used in EMG datasets, 8 features were chosen out of the 19 available features and they were obtained by applying an exhaustive (brute-force) feature selection technique for all feature sets. The technique is wrapper-based as classification accuracy was the criterion used to judge the

quality of a feature. Different classifiers were used and the other unselected 11 features always turned to be either redundant or irrelevant.
For the other three datasets, it is an ordinary spectral clustering problem and clustering is applied on all the data as there is no labelled part of the data. Spectral clustering (grouping) is evaluated here on its own and then its impact on classification performance is evaluated in Section 4.3.2. We are mainly interested in the proposed probabilistic threshold and probabilistic criteria similarity graphs so they are evaluated based on clustering evaluation measures compared to other similarity graph models existent in the literature.
In Table 4.1, values of the evaluation measures are shown for the following datasets:

- Abalone [Blake et al., 1998] dataset: 4177 instances, 9 features in 10 clusters.
- Swiss Banknotes [Flury and Riedwyl, 1983] dataset: 200 instances, 6 features in 2 clusters.
- Segmentation [Asuncion and Newman, 2007] dataset: 2310 instances, 19 features in 7 clusters.
- EMG Myopathic Upper Leg dataset (Myo Upper Leg): 557 instances, 8 features in 2 clusters.
- EMG Neurogenic Upper Leg dataset (Neuro Upper Leg): 356 instances, 8 features in 2 clusters.
- EMG Myopathic Lower Leg dataset (Myo Lower Leg): 583 instances, 8 features in 2 clusters.
- EMG Neurogenic Lower Leg dataset (Neuro Lower Leg): 444 instances, 8 features in 2 clusters.

The algorithms experimented are:

- Probabilistic threshold: optimal values of w and σ are obtained by cross-validation. There are two versions of the probabilistic thresholding approach as per how to transform the similarity matrix into a symmetric matrix:

  - Prob. thresholding Min.: Assign the minimum value out of $similarity(v_a, v_b)$ & $similarity(v_b, v_a)$ to both of them
  - Prob. thresholding Max: Assign the maximum value out of $similarity(v_a, v_b)$ & $similarity(v_b, v_a)$ to both of them

- Probabilistic criterion: optimal values of w and σ are obtained by cross-validation. There are two versions of the probabilistic acceptance criterion approach as per how to transform the similarity matrix into a symmetric matrix:
  - Prob. acceptance Min.: Assign the minimum value out of $similarity(v_a, v_b)$ & $similarity(v_b, v_a)$ to both of them
  - Prob. acceptance Max: Assign the maximum value out of $similarity(v_a, v_b)$ & $similarity(v_b, v_a)$ to both of them

- ϵ-neighbourhood: optimal value of ϵ is obtained by cross-validation.
- k-nearest neighbour: optimal value of k is obtained by cross-validation.
- Mutual k-nearest neighbour: optimal value of k is obtained by cross-validation.
- Fully connected graph: optimal value of σ is obtained by cross-validation.

The first 3 datasets are publicly available datasets that have been used in clustering before. The latter 4 datasets represent EMG datasets. As mentioned earlier, these datasets were acquired from upper leg and lower leg recordings and each of them contains two types of instances; normal as well as disordered (myopathic or neurogenic). In fact there are 2 rather than 4 EMG datasets as the myopathic and neurogenic upper leg datasets represent bags of the same dataset (the same goes for the lower leg dataset) but they are shown as two different datasets here because they are treated separately as far as spectral clustering (grouping) and its evaluation are concerned. In Section 4.3.2, where weakly supervised classification is applied, weakly annotated data of both upper leg datasets are being processed together along with the strongly labelled instances of the upper leg dataset (again the same goes for lower leg).

TABLE 4.1: Clustering indices values based on different similarity graph models.

Dataset	Algorithm	Eval. Index	Value
Abalone	Prob. threshold Min	F-measure	71%
	Prob. threshold Max		90.3%
	Prob. criterion Min		67.2%
	Prob. criterion Max		88.37%
	ϵ-neighbourhood		70.1%
	k-nearest neighbour		88.6%
	Mutual k-nearest neighbour		53.2%
	Fully connected graph		35.1%
Banknotes	Prob. threshold Min	F-measure	100%
	Prob. threshold Max		100%
	Prob. criterion Min		100%
	Prob. criterion Max		100%
	ϵ-neighbourhood		100%
	k-nearest neighbour		100%
	Mutual k-nearest neighbour		100%
	Fully connected graph		99%
Segmentation	Prob. threshold Min	F-measure	58.1%
	Prob. threshold Max		52.5%
	Prob. criterion Min		55.83%
	Prob. criterion Max		54.4%
	ϵ-neighbourhood		24%
	k-nearest neighbour		24.97%
	Mutual k-nearest neighbour		24.95%
	Fully connected graph		24.95%
Myo Upper Leg	Prob. threshold Min	Davies-Bouldin index	0.1402
	Prob. threshold Max		0.4137
	Prob. criterion Min		0.1400
	Prob. criterion Max		0.4132
	ϵ-neighbourhood		1.75
	k-nearest neighbour		2.27
	Mutual k-nearest neighbour		2.01
	Fully connected graph		1.4
Neuro Upper Leg	Prob. threshold Min	Davies-Bouldin index	0.6727
	Prob. threshold Max		0.3413
	Prob. criterion Min		0.6725
	Prob. criterion Max		0.3411
	ϵ-neighbourhood		2.39
	k-nearest neighbour		2.09
	Mutual k-nearest neighbour		2.11
	Fully connected graph		1.81
Myo Lower Leg	Prob. threshold Min	Davies-Bouldin index	0.2918
	Prob. threshold Max		0.3807
	Prob. criterion Min		0.2911
	Prob. criterion Max		0.3806
	ϵ-neighbourhood		1.18
	k-nearest neighbour		1.74
	Mutual k-nearest neighbour		1.71
	Fully connected graph		1.69
Neuro Lower Leg	Prob. threshold Min	Davies-Bouldin index	0.6521
	Prob. threshold Max		0.3642
	Prob. criterion Min		0.6520
	Prob. criterion Max		0.3641
	ϵ-neighbourhood		2.88
	k-nearest neighbour		2.97
	Mutual k-nearest neighbour		2.9
	Fully connected graph		2.04

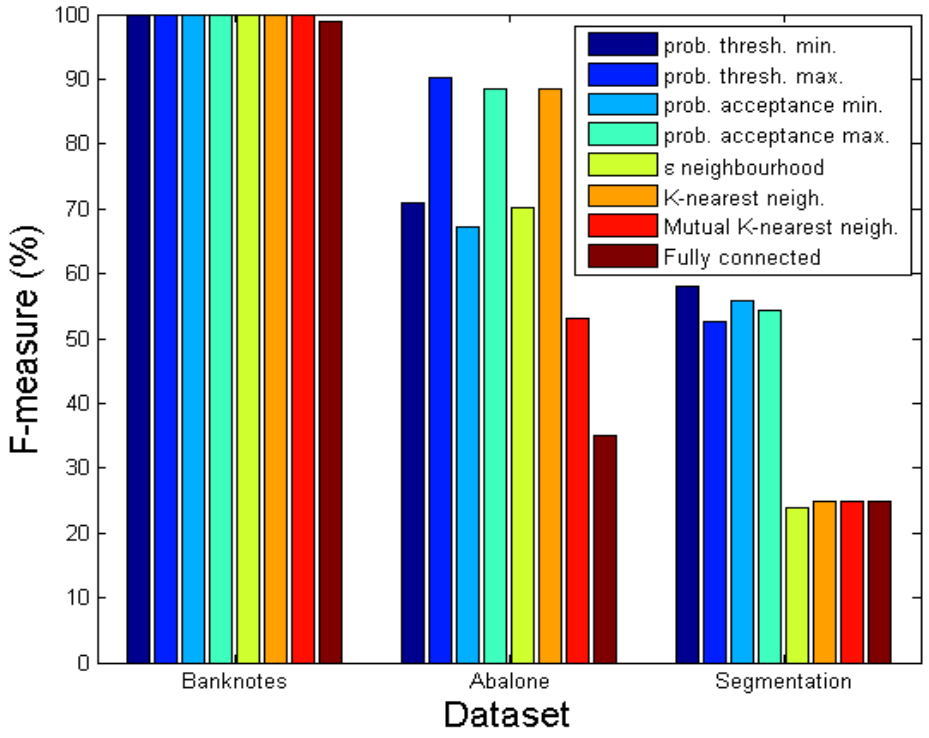


FIGURE 4.7: F-measure values for datasets with ground truth labels. The greater the F-measure value the better.

As can be seen in Table 4.1 and in figures 4.7 & 4.8, when similarity graph models are constructed using the proposed probabilistic approaches, clustering results are better, or at least as good as, the other approaches. Figure 4.8 shows the improvement achieved by using any of the four proposed similarity graphs over the other similarity graphs in comparison as the Davies-Bouldin index values are clearly better with the former. The same conclusion is shown in case of the Segmentation dataset (by far the largest out of the 3 datasets) in Figure 4.7. For Abalone dataset in Figure 4.7, the probabilistic threshold maximum graph leads to the best result as its F-measure value is slightly better than the one achieved by constructing the probabilistic criterion maximum graph as well as the K-nearest neighbour graph. All graphs are nearly equally good for the Banknotes dataset displayed in Figure 4.7. One other advantage of the proposed probabilistic thresholding approaches lies in the fact they do not depend on distance among the instances, location of the instances nor on

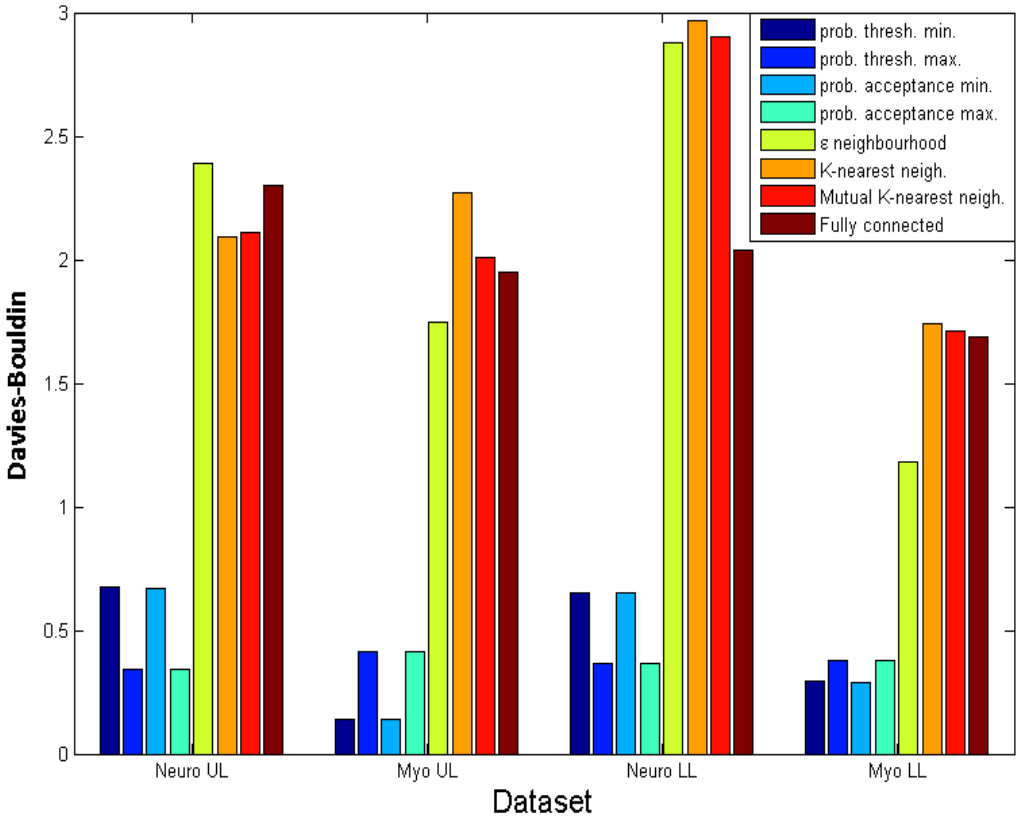


FIGURE 4.8: Davies-Bouldin index values for datasets without ground truth labels. The smaller the index value the better.

a number of neighbours specified a priori that can perform well at some part of the dataset but not on another part of the same dataset due to, for example, having a dataset containing different densities within it.

Results show that the probabilistic threshold similarity graph and the probabilistic criterion similarity graph lead to very similar clustering results among themselves as shown by the values of the validity indices. The former leads to better results in case of the Abalone dataset and the minimum graph of the Segmentation dataset, while the latter leads to slightly better results in case of the maximum graph of the Segmentation dataset. For the rest of the datasets, results are quite similar among the two proposed graphs.

On the other hand, for some datasets minimum similarity graphs lead to better results

than maximum similarity graphs and vice versa on the other datasets. This suggests that the choice between the minimum and maximum similarity graph for the same approach (probabilistic thresholding or probabilistic acceptance criterion) should depend on the value of the cluster validity index resulting from cross-validation.

4.3.2 Weakly Supervised Classification

Now that the annotated training data is larger due to the weak annotation of the previously unlabelled instances by spectral grouping, we want to evaluate the significance of the weak labelling procedure with respect to classification performance on the EMG muscle datasets. Results of the classification are evaluated before and after weak labelling, i.e. with a fully supervised classifier that assumes all training data instances can be directly assigned the label of their respective bags, and with a weak classifier that uses weakly labelled data resulting from spectral grouping along with strongly labelled data. The former is referred to as fully supervised in the sense that it assumes there is one level of data and assumes every instance has its own label, which turns it into a supervised learner. The assumption by which it casts the problem into a supervised learning problem and labels all instances, is invalid though. The latter classifier recognises the data as partially labelled and proceeds by the weak annotation and this is why it is referred to as a weakly supervised classifier.
A logistic regression classifier is used as the classification algorithm. Assuming F refers to a p-dimensional feature vector of an instance I, the utilised logistic regression classifier assumes $P(I = i|F) \propto \exp^{\beta_{i[0]}+\beta_{i[1:p]}f}, i < 3$.
The fully supervised classifier considers all training instances of myopathic and neurogenic muscles to have the same label (myopathic or neurogenic, respectively) as the muscle. On the other hand, the weak classifier exploits weak labels assigned to each of these instances by the spectral grouping procedure. The Upper Leg dataset has 650 labelled instances and 913 unlabelled instances. The fully supervised classifier imprudently assigns the respective bag label to 913 unlabelled instances. On the other hand, the weakly supervised classifier processes 913 weakly labelled instances as well as 650 strongly labelled instances. The Lower Leg dataset has 672 labelled instances and 1027 unlabelled instances. Therefore,

TABLE 4.2: Muscle classification accuracy based on the proposed weakly supervised classifiers vs. a fully supervised classifier.

Dataset	Algorithm	Classification accuracy
Upper Leg	Prob. threshold Min Weak Class.	95%
	Prob. threshold Max Weak Class.	92.4%
	Prob. criterion Min Weak Class.	95%
	Prob. criterion Max Weak Class.	92.7%
	Fully Supervised Class.	84%
Lower Leg	Prob. threshold Min Weak Class.	96.1%
	Prob. threshold Max Weak Class.	93.4%
	Prob. criterion Min Weak Class.	95.6%
	Prob. criterion Max Weak Class.	93.4%
	Fully Supervised Class.	82.3%

the fully supervised classifier imprudently assigns the respective bag label to 1027 instances whereas the weakly supervised classifier processes 1027 weakly labelled instances as well as 672 strongly labelled instances.

Leave-one-out cross-validation is implemented by setting instances belonging to a single muscle as test data while training on instances of the rest of the muscles, then repeating this process for every muscle. It can be more precisely referred to here as leave-one-muscle-out cross-validation. Overall muscle classification accuracy is the main metric used to evaluate the classification performance. Table 4.2 shows the results of the logistic regression fully supervised and weakly supervised classifiers. Every weak classifier is named after the similarity graph model pursued but a logistic regression classifier is used for classification in all weak classifiers as well as the supervised classifier. Results show that, using a weak classifier, muscle classification accuracy significantly improves compared to the fully supervised classifier.

4.4 Summary

A weakly supervised learning paradigm is introduced. The goal is to improve classification performance by weakly annotating unlabelled data using a spectral graph-theoretic grouping

strategy. Spectral grouping exploits similarity among data instances as well as the relationship between unlabelled and strongly labelled data instances, by constructing similarity graph models, including two introduced similarity graph models, to weakly annotate unlabelled data instances. Afterwards, a classifier learns from the weakly and strongly labelled data. Datasets upon which applying the introduced weakly supervised learning paradigm may be useful, typically consist of a limited number of labelled samples and a larger set of unlabelled samples as well as intrinsic relationship between both sets derived by the data or the problem. Results show that performance of the resulting weakly supervised classifier is better than its counterpart fully supervised classifier on the EMG datasets.

In addition to improving the classification performance on partially labelled data, spectral graph-theoretic grouping using the two introduced similarity graph models can be seen as a spectral clustering algorithm with the goal in this case being to create clusters with high similarity within each cluster and low similarity between clusters. Results show that the introduced spectral grouping strategy leads to compact and well separated clusters on the tested datasets.

The introduced weakly supervised learning paradigm is discriminative. Therefore, data can not be simulated as this discriminative paradigm does not model the joint distribution of data and label. One of the future goals related to EMG data is to build a simulation tool. A generative model is needed for this to be achievable.

Another limitation of the weakly supervised learning paradigm is that spectral graph-theoretic grouping can not handle all bags-of-instances datasets. In the case of EMG, spectral graph-theoretic grouping makes it possible to exploit unlabelled data in learning, but this is due to the assumptions of the EMG data that make clustering a sound option. However, with the MUSK dataset, where many positive bags have a very small number of positive instances that can not be considered a group or a cluster, the weakly supervised learning paradigm does not represent good approach.

Next chapter we introduce a generative multiple-instance learning (MIL) modelling paradigm that is intuitively capable of handling different assumptions of bags-of-instances datasets, as well as making use of these assumptions in learning.

Chapter 5

Generative Multiple-Instance Learning Models

5.1 Motivation

In Multiple-Instance Learning (MIL), training instances are grouped together in bags which have labels. Each instance in a bag has a label that may be different from that of the bag, but instance labels are not observed; only the label of the bag is available for learning [Adel et al., 2013]. The MIL setting was first proposed by Dietterich et al. [Dietterich et al., 1997] while dealing with a pharmaceutical problem. Their task was to predict the binding properties of molecules, which depend on the shape of the molecule. However, a molecule can take on several shapes. Thus, each molecule is represented as a bag of instances, whereas each instance represents a shape the molecule can take on. If none of the possible shapes enable binding, the bag (molecule) gets a negative label. But if one shape allows for binding, the bag is labelled positive. The dataset from this problem is referred to in MIL literature as the MUSK dataset [Dietterich et al., 1997]. The MUSK dataset from this problem has remained one of the most widely used benchmark datasets for MIL tasks.

The classification of a muscle based on the set of MUPTs representing a sampling of its MUs can be formulated as an MIL problem wherein each muscle is a bag and each MU of a muscle is an instance of that bag. In this chapter, generative modelling approaches

are shown to be useful and effective for data that naturally occur in MIL form, such as EMG muscle data. Generative models are models that describe the full joint distribution of the data [Adel et al., 2013]. This is one of the main differences between the weakly supervised learning paradigm presented in Chapter 4 and the paradigm presented in this chapter, as the former is discriminative. Predicting with a generative model is particularly suitable for medical domains for several reasons: Generative models allow for expert domain knowledge to be incorporated in an intuitive way, which leads to good inductive bias in the modelling assumptions. As we will demonstrate, a model with good inductive bias (elicited from experts in biomedicine) can result in highly accurate predictions even on the basis of a relatively small training set. Generative models yield not only a classification tool, but can also be used as a simulation tool for the problem domain.

For EMG muscle datasets, the features of an instance correspond to the features used to represent the MUPT of the MU. Main assumptions of the EMG problem are as follows. The instances of a normal bag are all normal, while neurogenic and myopathic bags may contain both normal as well as neurogenic or myopathic instances, respectively. It is exceedingly unlikely that a neurogenic (resp. myopathic) disordered muscle contains/generates a myopathic (resp. neurogenic) MU/MUPT. Only bag labels are provided in the training dataset. A learning experiment was performed where a learner assigned the bag label to each instance in the bag and it proved to be inefficient. There is a need for a learner that learns instance labels of the training set first, then learns to classify test instances (for the sake of classifying their respective bags) and test bags accordingly.

The generative models introduced in this chapter not only provide a solution to the muscle classification problem but also provide a framework as well intuition and guidelines for applying generative models to other MIL problems. We provide the results of applying the generative modelling approaches on the EMG muscle data, and also on the MUSK dataset. Most of the content of this chapter is presented in Adel et al. [2013].

5.2 Related Work

Dietterich et al. [1997] suggest several algorithms for learning axis-aligned rectangles for the original MIL problem on the MUSK data. Following Dietterich's introduction of the MUSK data setting as MIL, various problems have been phrased as MIL problems. MIL approaches have, for example, been employed for content-based image retrieval [Maron and Ratan, 1998, Zhang et al., 2002], text classification [Settles et al., 2007, Andrews et al., 2002b], protein identification [Tao et al., 2004], music information retrieval [Mandel and Ellis, 2008] and activity recognition [Stikic and Schiele, 2009]. In medical domains, prediction problems often naturally occur as MIL tasks. One example is the original MUSK prediction task; Dundar et al. [2008] also show that learning problems for computer-aided detection (CAD) applications can often be considered as MIL problems.

Wang and Zucker [2000] adapt Nearest Neighbour learning to MIL. Several studies present kernels to use Support Vector Machines on MIL problems [Gärtner et al., 2002, Andrews et al., 2002a, Tao et al., 2004], adaptations of boosting [Andrews and Hofmann, 2003, Xu and Frank, 2004, Viola et al., 2005] and, more recently, incorporate methods from semi-supervised or active learning to the MIL setting [Rahmani and Goldman, 2006, Zhou and Xu, 2007, Settles et al., 2007]. The original bag labelling rule (where the label of the bag is the logical OR of its instances as in the MUSK data) has been modified and generalized to account for requirements by other application areas (Foulds and Frank [2010] provide an overview.). Sabato et al. [2010] provides a comprehensive study with upper and lower bounds on the sample complexity of the bag-level learning problem without the independence assumption, i.e. without assuming that instances occurring together in a bag are conditionally independent .

Generative modelling approaches have only rather recently been introduced to the MIL setting [de Freitas and Kück, 2005, Yang et al., 2009, Foulds and Smyth, 2011]. The former two studies suggest more complex model structures for modifications of the Multiple-Instance Learning problem. The works of Foulds and Smyth [2011] and Yang et al. [2009] can be viewed as special cases of the paradigm introduced in this chapter. Regarding references related to graphical models, in Alpaydin [2010] Alpaydin gives a concise overview of directed

graphical models while in Koller and Friedman [2009] Koller and Friedman provide a comprehensive treatment of the subject. Also in Chapter 6, we inspect graphical models a bit more deeply.

5.3 MIL Generative Models

Sticking to the mathematical notation introduced in Chapter 3. Random variables are denoted in upper case, while their realizations are denoted in lower case. Let $t = 3$ be the number of possible bag/instance labels. Let $B \in \{1, 2, ..., t\}$ be the random variable representing a bag's label, and let $I_j \in \{1, 2, ..., t\}$ be the random variable representing the label of the jth instance belonging to the bag. The number of instances in the bag is denoted by m; the m instance labels are referred to together as the vector $\vec{I}$. Let $\vec{F}_j \in \mathbb{R}^p$ be the p-dimensional feature vector belonging to the jth instance. Elements of a vector are indexed with a square-bracketed subscript, so $\vec{F}_{j[k]}$ is the kth element of the observed feature vector of the jth instance in a bag. A "generic" instance label is referred to as I and a "generic" feature vector is referred to as F.

Most MIL work to date considers binary labels, i.e. $t = 2$, while in this EMG problem $t = 3$, since a bag or instance can be either *normal* ($B = 1$), *myopathic* ($B = 2$), or *neurogenic* ($B = 3$). Furthermore, in this problem, a bag may only generate a *compatible* instance label: the value of a single instance label i_j is called *compatible* with a bag label b if and only if $i_j \in \{1, b\}$. A joint labelling $\vec{i} = i_1, ..., i_m$ is called *feasible* if and only if $\exists b(\forall j, i_j \in \{1, b\})$.

The following is a discussion of possible Bayes net structures for the proposed MIL generative models followed by a discussion of the possible modelling choices for marginal and conditional distributions.

5.3.1 Model Structures

The main inductive bias that is retained here from the original MIL formulation is the assumption that the bag label is conditionally independent of the feature vectors given their corresponding instance labels. This is implied by the assumption that the feature probability

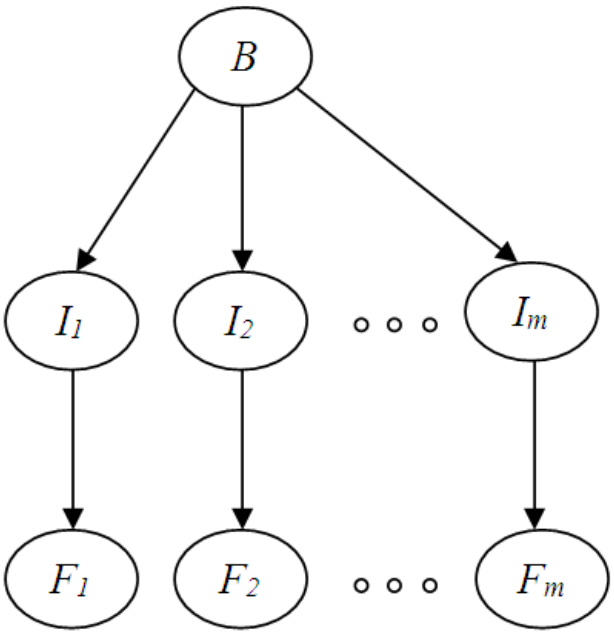


FIGURE 5.1: The BIF model structure. Parameters for $P(I_j|B)$ and for $P(\vec{F}_j|I_j)$ are tied across j.

distribution of normal instances in a normal bag should be the same as the feature probability distribution of normal instances in an abnormal bag. The same assumption applies to the MUSK dataset. Three possible Bayes net structures are presented below, two of which will be used as candidate structures for the the EMG MIL generative model. For completeness, a fourth structure that does not satisfy the conditional independence assumption is also discussed. The structures are named based on the partial order in which the variables appear in the graph.

5.3.1.1 BIF: $B \longrightarrow \boxed{\rightarrow I \rightarrow F \ _m}$

This structure best represents the generative process underlying the EMG data. Under this structure, the bag generates its m instances independently given its label. Each instance in turn generates its own feature vector given *its* label, but independent of the bag label and independent of the other instances and features in the bag. The structure of this model is given concisely by the plate diagram $B \longrightarrow \boxed{\rightarrow I \rightarrow F \ _m}$, the expanded version of which is shown in Figure 5.1. The other model structures considered differ in the direction of the edges, which has important consequences as will be seen.

For this structure, $P(B)$ must be learnt from observed data, while $P(I|B)$ and $P(F|I)$ must be learnt using a hidden variable method like EM. Constraints on $P(I|B)$ are simple to encode in order to ensure that a bag can *never* generate an incompatible instance. The values in the conditional probability table (CPT) of $B|I$ can be restricted by requiring $P(I = i|B = b) = 0$ for all $i \notin \{1, b\}$. It follows that sets of instance labels drawn from this model are always compatible with the bag label. Furthermore, a Dirichlet prior can be imposed easily on the proportion of instance labels that match the bag label while obeying these constraints. Since the continuous features are observed, $P(F|I)$ is modelled using density estimation.

The main departure this model makes from other probabilistic models for multiple instance learning is that it assumes that the bag label is the cause of the instance label, rather than the other way round. Under this model, it is possible for a disordered (non-normal) bag to produce all normal instances, which is disallowed in other MIL models. However, for this EMG system, this is entirely appropriate; it is possible (though unlikely) for a muscle with a myopathic or neurogenic disorder to produce all normal MUPTs. Among existing MIL models, that of Yang et al. Yang et al. [2009] is most similar to BIF.

5.3.1.2 FIB: $B \leftarrow \boxed{I \leftarrow F}_{m}$

FIB represents another way of expressing an MIL generative model where the instances generate the bag label. Under this structure, the feature vectors are drawn from some $P(F)$, they then generate the instance labels, which in turn determine the bag label. The probability $P(I|F)$ can be expressed using any discriminative learner, which is attractive, though a hidden variable method like EM must still be used for training. In order to make the model fully generative $P(F)$ must also be modelled using density estimation.

Note that B has all m instances as parents, and m can vary from muscle to muscle, so $P(B|I)$ must be expressed as a function of different numbers of instances; this is discussed in Section 5.3.2. Unfortunately, in the EMG 3-class setting, this structure suffers from an important drawback: it offers no way of prohibiting infeasible instance label assignments, i.e. assignments where for example $I_1 = 2$ and $I_2 = 3$. In order to have a fully consistent

generative FIB model, therefore, an additional possible bag "label" $b = 0$ that has positive probability given infeasible labellings, must be added. This does not reflect the generative process of the EMG data, but we can still use this model structure and condition on the event $B \neq 0$ where necessary [Adel et al., 2013]. Foulds and Smyth [2011] note that prior knowledge about the frequency of instance labels given bag labels is difficult to incorporate with directed edge from I to B.

5.3.1.3 IBF: $B \leftarrow \boxed{I \rightarrow F}_m$

Under this structure, the instances are generated independently according to some $P(I)$, and they subsequently generate both the bag label and the feature vectors. This model structure is essentially the "Multiple-Instance Mixture Model" of Foulds and Smyth [2011]. As in the FIB model, there is a directed edge from I to B, which causes the same drawbacks described in the FIB model but does *not* allow for the additional flexibility of using a discriminative learner for the instance labels. Therefore, this structure is not considered for the EMG MIL generative model.

5.3.1.4 Alternative Model BFI: $B \rightarrow \boxed{I \leftarrow F}_m$

This model attempts to combine two attractive properties: The ability to use a discriminative model $P(I|F)$, and the ability to easily put priors on $P(I|B)$. Unfortunately, in this model, B and F are dependent given I, which is not true in this EMG problem (nor with the MUSK data). Note that BFI is essentially a clustering model with I as the cluster label and B acting simply as an additional feature along with F. No particular extra importance is attached to B via conditional independence so some additional assumption about $P(I|B, F)$ is needed. Put another way, even if the feature label is known, the bag label exerts an influence on the instance feature vector. This means that normal instances from B=1 are different from normal instances in B=2, which is not correct. We count on being able to generalize from normal instances in normal bags to normal instances in abnormal bags, so this model is not appropriate.

5.3.2 Model Components

Choosing the model structure determines the conditional independence properties of the MIL generative model but does not specify a form for the various probability distributions. In this section, we discuss some possibilities for components of the model that have different assumptions and inductive biases.

5.3.2.1 $P(B)$ and $P(I|B)$ for the BIF Structure

Since B and I take on a small number of discrete values, a tabular representation is appropriate. As noted earlier, we can impose restrictions on possible values of $I|B$ by clamping appropriate values in the table.

5.3.2.2 $P(F|I)$ for the BIF Structure

Because a continuous feature space is assumed, $P(F|I)$ can be modelled using any density estimation method. Some well-known possibilities are discussed here.

Multivariate Gaussian One simple choice for $P(F|I = i)$ is a multivariate Gaussian distribution with mean μ_i and covariance Σ_i for $i \in \{1, ..., t\}$. Depending on the availability of data and desired modelling assumptions, we can restrict Σ_i to be diagonal.

Kernel Density Estimation Kernel Density Estimation (KDE) is a non-parametric method that estimates a probability density or distribution function by summing up *kernel* functions placed at every observed data point. The most common form of KDE, which uses a Gaussian kernel, is used. To choose the kernel width, the *Silverman's rule of thumb* Silverman [1998] ($h = (\frac{4\times\hat{\sigma}^5}{3n})^{\frac{1}{5}}$; n = *number of samples*, $\hat{\sigma}$ *is the standard deviation of the samples*) is picked as a simple but effective choice. The advantage of KDE is that it is capable of modelling complex marginals *and* complex dependencies among the variables of interest, but it does not always work well in moderate to high dimensions.

Gaussian Copula with KDE Marginals A drawback of the multivariate Gaussian approach is that much real-world data is not in fact Gaussian. Since it is required to make this generative model as realistic as possible, a copula-based model is proposed. This copula-based model is practical to estimate but can fit the observed data more closely.

Sklar's theorem [Sklar, 1959] implies that any multivariate density g with marginal densities $g_1, g_2, ..., g_p$ and marginal cumulative distribution functions (CDFs) $G_1, G_2, ..., G_p$ can be expressed in the form

$$g(f) \quad = \quad g_1(f_{[1]}) \cdot g_2(f_{[2]}) \cdot ... \cdot g_p(f_{[p]}) \cdot c(G_1(f_{[1]}), G_2(f_{[2]}), ..., G_p(f_{[p]})) \quad (5.1)$$

where c is a *copula density* that captures the dependence structure of the feature vector $F = (F_{[1]}, ..., F_{[p]})$. If the elements of F are independent, then $c \equiv 1$.

Because they are all one-dimensional, the marginals can be estimated well using Kernel Density Estimation (KDE) Alpaydin [2010] even with a modest amount of data, giving $\hat{g}_k$ and $\hat{G}_k$ for $k = 1...p$. This allows the MIL generative model to capture non-Gaussian aspects of the data, such as relatively heavy or light tails, skewness, or even multi-modality, thus making it more realistic.

The copula model allows to achieve more high-fidelity marginals without resorting to an unrealistic independence assumption: we can still capture pairwise dependencies in the data by assuming a parametric form for c and estimating the necessary parameters. A Gaussian copula is assumed, whose parameter is the covariance matrix of
$(\Phi^{-1}(G_1(F_{[1]})), \Phi^{-1}(G_2(F_{[2]})), ..., \Phi^{-1}(G_p(F_{[p]})))$ where Φ^{-1} is the inverse of the standard normal CDF. This can be estimated by the empirical correlation of
$(\Phi^{-1}(\hat{G}_1(F_{[1]})), \Phi^{-1}(\hat{G}_2(F_{[2]})), ..., \Phi^{-1}(\hat{G}_p(F_{[p]})))$ over the observed $f|I = i$ in the data. (a separate copula model is estimated for each possible value of I).

5.3.2.3 $P(F)$ for the FIB Structure

In principle, any of the density estimators proposed for $P(F|I)$ could be used here; however, the marginal $P(F)$ is multi-modal, so the copula and KDE models are more appropriate as

density estimators in this model.

5.3.2.4 $P(I|F)$ for the FIB Structure

Since I has a discrete domain, any classification method that supplies class probabilities can be used to model $P(I|F)$. We examine three such methods here.

Logistic Regression This well-known model assumes $P(I = i|F) \propto \exp^{\beta_{i[0]}+\beta_{i[1:p]}f}, i < t$. Note that, a maximum likelihood estimate of β is not unique if the data is linearly separable.

Support Vector Machines (SVM) SVM classifiers have the added advantage that in the event of feature separability, one gets a large-margin classifier whereas logistic regression would fail to converge. Furthermore, kernelized SVM classifiers allow us to easily create non-linear separators in feature space.

The dual form of the soft margin version of SVM is given by:

$max \;\; \Sigma_{u=1}^{N}\alpha_u - \frac{1}{2}\Sigma_{u,v=1}^{N} I_u I_v \alpha_u \alpha_v k(\vec{f_u}\vec{f_v})$

$w.r.t. \;\; \alpha_u$

$s.t. \quad 0 \leq \alpha_u \leq C$

$and \quad \Sigma_{u=1}^{N}\alpha_u I_u = 0$

where N refers to the total number of instances, k refers to a kernel function and C is a penalty parameter that controls overfitting. If $C = 0$, then there is no penalty and a maximum margin is obtained. If C is large, it results in fewer classification training errors and a smaller margin. Here a radial basis kernel $k(\vec{f_u}, \vec{f_v}) = exp(-\gamma||\vec{f_u} - \vec{f_v}||^2)$ is used with $\gamma = \frac{1}{8}$ and a penalty parameter $C = 1$.

Multi-class SVM is implemented using a one-against-one strategy. A voting strategy is used for each instance where each binary classification result is a vote. The assigned class is the one with the maximum number of votes.

Although the classic SVM formulation [Cortes and Vapnik, 1995] does not provide conditional class probabilities, subsequent work [Huang et al., 2006, Chang and Lin, 2011] has added this capability.

In the beginning pairwise class probabilities are estimated as $r_{uv} \approx p(I = u | I = u \text{ or } v, \vec{f})$ [Chang and Lin, 2011]. After estimating all r_{uv} values, values of p are obtained by solving an optimisation problem, which is [Wu et al., 2004]:

$min_p \quad \frac{1}{2}\Sigma_{u=1}^{t}\Sigma_{v:v\neq u}(r_{vu}p_u - r_{uv}p_v)^2$

$s.t. \quad p_u \geq 0, \forall i,$

$and \quad \Sigma_{u=1}^{t} p_u = 1$

K-Nearest Neighbours If the decision boundary between instance labels is believed to be complex and if sufficient data are available, a non-parametric model may be warranted. K-nearest neighbours uses the empirical distribution of the instance labels of the K closest feature vectors to f to estimate $P(I|F = f)$.

5.3.2.5 $P(B|I)$ for the IBF and FIB Structures

Since m varies from bag to bag, $P(B|I)$ must be expressed as a function that can take a variable number of parameters. Recall that in this setting, we must allow for the possibility that the joint labelling of I is not feasible; $b = 0$ is added to the domain of B to capture this event. we can adhere to the standard MIL assumptions by making $P(B|I)$ deterministic as follows. For feasible labellings, we set

$$P(B = b|I_1, I_2, ..., I_m) = \begin{cases} 1 & \text{if } b = \max_j I_j \\ 0 & \text{otherwise,} \end{cases} \tag{5.2}$$

and for infeasible labellings we set

$$P(B = b|I_1, I_2, ..., I_m) = \begin{cases} 1 & \text{if } b = 0 \\ 0 & \text{otherwise.} \end{cases} \tag{5.3}$$

This means that, for EMG datasets, a bag whose instances form a feasible joint labelling has a label equal to 1 if all its elements are normal ($\max_j I_j = 1$), otherwise it has either

Algorithm 1 Hard EM Algorithm

for all bags **do** {initialise instance labels}
  $\vec{i} \leftarrow b$
end for
repeat
  learn model components {M-step}
  for all bags **do** {relabel instances: E-step}
    $\vec{i} \leftarrow \text{argmax}_{\vec{i}} P(\vec{I} = \vec{i} | B = b, \vec{F} = \vec{f})$
  end for
until instance labels do not change

label 2 or 3. Note that a feasible joint labelling with labels 2 and 3 both happening does not exist as this is an infeasible labelling.

5.4 Learning and Inference

All of the components described in Section 5.3.2 have associated off-the-shelf learning algorithms for completely observed data. MIL generative models must be learnt without ever observing I (though with substantial information about I provided through B), so a hard-Expectation-Maximization (EM) procedure is used for simultaneously learning the model parameters and inferring the most likely I given the observed B and F. This worked well on the EMG data. Moreover, the conceptual groundwork that is laid here may also be useful in other settings.

5.4.1 Learning

For learning, a "hard-EM" approach [Koller and Friedman, 2009] is used. As noted in Chapter 3, access to a collection of n bags of the form $(b, f_1, f_2, ..., f_{m_\nu})_\nu, \nu \in \{1, ..., n\}$ which are independent and identically distributed (i.i.d), is assumed. Given an initial label assignment to all of the instances in the EMG dataset, the learning method is straightforward. Steps of the learning algorithm are presented in Algorithm 1. The following is a discussion of the two main steps.

5.4.1.1 Parameter Estimation

BIF For the BIF model, $P(B)$, $P(\vec{I}|B)$, and $P(F|I)$ must be estimated. The marginal probability $P(B)$ is estimated from observed bag label counts only; it does not change across iterations. Because we assume $P(I|B)$ is the same for all instances in all bags, we pool all the bags together and use the aggregated counts to estimate $P(I|B)$. We may add "pseudo-counts" to this estimate if a dirichlet prior is desired; in these experiments we assume for each bag type that each compatible instance label has been seen once, and each incompatible label zero times. To learn $P(F|I)$, again we may pool all of the instances together to learn d density estimates $P(F|I=1), P(F|I=2), ..., P(F|I=t)$.

FIB For the FIB model, we must estimate $P(F)$ and $P(I|F)$; we assume that $P(B|I)$ is fixed according to the standard MIL definition. To estimate $P(F)$, we pool all feature vectors together and estimate the necessary parameters. These are completely observed so $P(F)$ does not change across iterations. To learn $P(I|F)$, again we pool all of the instances together to learn the conditional distribution using a supervised learning method.

5.4.1.2 Label Updating

To update the labels for each bag given the learned parameters, we must find the most likely instance labels given the observed data, that is, we must compute $\text{argmax}_{\vec{i}} P(\vec{I} = \vec{i}|B = b, \vec{F} = \vec{f})$ for each bag.

BIF From the conditional independence structure of the BIF model, we have

$$\underset{\vec{i}}{\text{argmax}}\, P(\vec{I} = \vec{i}|B = b, \vec{F} = \vec{F}) = \underset{\vec{i}}{\text{argmax}}\, P(\vec{I} = \vec{i}|B = b)P(\vec{F} = \vec{f}|\vec{I} = \vec{i}).$$

Since these quantities for different instances are independent given the bag label, we have

$$P(\vec{I} = \vec{i}|B = b)P(\vec{F} = \vec{f}|\vec{I} = \vec{i}) = \prod_{j=1}^{m} P(I_j = i_j|B = b)P(F_j = f_j|I_j = i_j),$$

so to maximize the likelihood of the joint label assignment $\vec{i}$, we may maximize each instance label independently:

$$\vec{i}_{[j]} = \underset{i_j}{\operatorname{argmax}} P(I_j = i_j | B = b) P(F_j = f_j | I_j = i_j).$$

FIB From the conditional independence structure of the FIB model, we have

$$\underset{\vec{i}}{\operatorname{argmax}} P(\vec{I} = \vec{i} | B = b, \vec{F} = \vec{F}) = \underset{\vec{i}}{\operatorname{argmax}} P(B = b | \vec{I} = \vec{i}) P(\vec{I} = \vec{i} | \vec{F} = \vec{f}).$$

However, in this model, the instance labels are *not* conditionally independent given the bag label and we cannot maximize them independently. For example, if $B = 2$, and $I_1, I_2, ..., I_{m-1}$ are all equal to 1, then I_m must be equal to 2 according to the MIL assumption encoded in $P(B|\vec{I})$. However, we can still avoid searching over all m^t possible label vectors. If and only if $\vec{i}$ is a feasible vector for $B = b$, then we have $P(B = b|\vec{i}) = 1$ and therefore the objective is

$$P(B = b|\vec{i}) P(\vec{I} = \vec{i} | \vec{F} = \vec{f}) = \prod_{j=1}^{m} P(I_j = i_j | F_j = f_j).$$

We can find the best feasible $\vec{i}$ in two steps:

1. Let $\vec{i_j} = \operatorname{argmax}_{i_j \in \{1,b\}} P(I_j = i_j | F_j = f_j)$
2. If $\vec{i} = \vec{1}$, set the label for index given by $\operatorname{argmax}_j P(I_j = b | F_j = f_j)$ to b.

The vector $\vec{i}$ is now feasible and maximizes $\prod_{j=1}^{m} P(I_j = i_j | F_j = f_j)$ over all feasible vectors when $B = b$.

5.4.2 Inference

Once we have learned all of the model parameters, given a new bag where only feature values $\vec{f}$ are observed, we wish to compute $\operatorname{argmax}_{\vec{i},b} P(I = \vec{i}, B = b | \vec{F} = \vec{f})$. These are the most likely bag and instance labels given the m feature vectors in the new bag.

BIF In the BIF model,

$$\max_{\vec{i},b} P(I=\vec{i}, B=b|\vec{F}=\vec{f}) \propto \max_{\vec{i},b} P(B=b)P(\vec{I}=\vec{i}|B=b)P(\vec{F}=\vec{f}|\vec{I}=\vec{i})$$
$$= \max_{b} P(B=b) \max_{\vec{i}} P(\vec{I}=\vec{i}|B=b)P(\vec{F}=\vec{f}|\vec{I}=\vec{i}).$$

Therefore we can apply the instance label updating method presented in Section 5.4.1 for each possible bag label and weight them according to $P(B)$ to find the joint MAP assignment to b and $\vec{i}$.

FIB In the FIB model,

$$\operatorname*{argmax}_{\vec{i},b} P(I=\vec{i}, B=b|\vec{F}=\vec{f}) = \operatorname*{argmax}_{\vec{i},b} P(B=b|\vec{I}=\vec{i})P(\vec{I}=\vec{i}|\vec{F}=\vec{f})$$
$$= \operatorname*{argmax}_{b} \operatorname*{argmax}_{\vec{i}} P(B=b|\vec{I}=\vec{i})P(\vec{I}=\vec{i}|\vec{F}=\vec{f}).$$

Therefore we can apply the instance label updating method presented in Section 5.4.1 for each possible bag label to find the joint MAP assignment to b and $\vec{i}$.

5.5 Experiments

We find that the BIF structured models perform very well for several different component choices. The FIB structured models perform less well, but still much better than chance as per bag accuracy. Based on the results obtained, we recommend structure and component choices that lead to a high-fidelity generative model of EMG data.

5.5.1 EMG Datasets

We use $p = 8$ features, which are the same 8 features used in the weakly supervised paradigm described in Chapter 4. The same 8 features were also used in prior work [Adel et al., 2012] as they were found to be useful for classification. Descriptions of these features are provided

in Table 3.1.

1. **Number of turns**
2. **Amplitude**
3. **Area**
4. **Thickness**
5. **Size index**
6. **Turn width**
7. **Firing rate MCD (mean consecutive difference)**
8. **A-jiggle**

Two EMG datasets are used here. One is acquired from upper leg recordings and it contains 99 bags and 1563 instances. The other EMG dataset is acquired from lower leg recordings and it contains 95 bags and 1699 instances. All data were collected under IRB approval and and were de-identified. Versions used here have been cleaned to remove obvious outlier errors. For example, instances with highly improbable feature values were removed.

5.5.2 Results on EMG Datasets

Table 5.1 shows the performance of different models on the EMG data. The accuracy results were computed using leave-one-bag-out cross-validation. We also present the log-likelihood of the observed data maximized over the model parameters and the hidden instance labels, which measures how well the models fit the training data.

We present the results for the BIF structure using five different density estimators for $P(F|I)$. We use two versions each of the Gaussian and copula models, one assuming independence between elements of the feature vectors given the instance labels (e.g. a diagonal covariance matrix) indicated by the prefix $\perp\!\!\!\perp$, and one assuming pairwise correlations. The marginals of the copula models are estimated using KDE with a Gaussian kernel and the Silverman's rule of thumb bandwidth [Silverman, 1998]. We also give results for a multi-dimensional KDE for $P(F|I)$, again with the Silverman's rule of thumb bandwidth.

As mentioned earlier, a tabular representation is used in learning the BIF structure in order to initiate $P(I|B)$ where restrictions are imposed on certain values of $I|B$ by clamping these values in the table. More specifically, the following values in the table were assigned 0 values: $P(I = 2|B = 1)$, $P(I = 3|B = 1)$, $P(I = 3|B = 2)$ & $P(I = 2|B = 3)$. Table 5.2 shows the accuracy and likelihood values obtained when a tabular representation that assumes small values (0.01) rather than zeros in the aforementioned table positions and the BIF structure was robust enough to produce similar results to those obtained by setting the same values to 0.

Results for the FIB structure are presented using four different discriminative learning models. In all cases, $P(F)$ was estimated using a multi-dimensional KDE with the Silverman's rule of thumb bandwidth. The discriminative learners were Logistic Regression (LR), K-nearest neighbours (KNN) with $K = 7$, SVMs with a radial basis function kernel, $C = 1$ and $\gamma = 1/8$, and Quadratic Discriminant Analysis (QDA). QDA assumes a Gaussian distribution of each label/class and assigns to an instance the class with a greater value of posterior probability [Hastie et al., 2009]. Parameters of the Gaussian distribution of each class are estimated from the training data by a maximum likelihood estimate (MLE). A posterior probability refers to $p(I|F)$, and the assigned label is obtained by $i = argmax_i\ p(F|I)(p(I)$. KNN suffers due to not having sufficient data instances for it to perform well. Also, performance of SVM is very bad.

The last column shows the previous state of the art results, which were obtained without the bag-instance assumption. Bag labels were given as the label of each instance within the bag and QDA was used assuming independence between features.

5.5.3 Results on the MUSK Dataset

The MUSK dataset [Dietterich et al., 1997] contains 92 bags and 476 instances. Each instance has 168 features. All feature values are discrete. Version 1 of the MUSK dataset is used and it is referred to as MUSK1. Table 5.3 shows results on the MUSK1 dataset. We use the same models. Since the data are 168-dimensional, as a pre-processing step, we use PCA to eliminate near-collinearity; we choose enough components to capture 90% of the

TABLE 5.1: Results of the EMG datasets. To give a sense of the statistical uncertainty, we mark all accuracies that are within the 99% Bernoulli confidence interval of the maximum observed accuracy in bold. We mark the highest log-likelihoods for the BIF and FIB structures in italics.

Upper Leg	BIF: $B \longrightarrow I \rightarrow F$ $_m$					FIB: $B \longleftarrow I \leftarrow F$ $_m$				Non-MIL
	⫫Gauss	⫫Cop.	Gauss	Cop.	KDE	LR	KNN	SVM	QDA	⫫QDA
Bag Acc.	**95.5**%	**95.5**%	**95.5**%	**95.5**%	**95.5**%	72.8%	56.8%	25.0%	**83.8**%	81.1%
Log Lik.	-36843	-37104	-36810	-37066	*-34726*	*-32889*	-32998	-34382	-32938	—
Lower Leg	BIF: $B \longrightarrow I \rightarrow F$ $_m$					FIB: $B \longleftarrow I \leftarrow F$ $_m$				Non-MIL
	⫫Gauss	⫫Cop.	Gauss	Cop.	KDE	LR	KNN	SVM	QDA	⫫QDA
Bag Acc.	**98.6**%	**97.1**%	**97.1**%	**95.7**%	88.6%	81.4%	58.6%	37.1%	82.6%	77.1%
Log Lik.	-38035	-38206	-37980	-38141	*-35999*	*-34833*	-34952	-35598	-35128	—

TABLE 5.2: Results of the EMG datasets with the BIF structure using different initial values of $P(I|B)$.

Upper Leg	BIF: $B \longrightarrow I \rightarrow F$ $_m$				
	⫫Gauss	⫫Cop.	Gauss	Cop.	KDE
Bag Acc.	95.1%	**95.1**%	**95**%	**95.1**%	**95**%
Log Lik.	-36843	-37105	-36810	-37068	-34726
Lower Leg	BIF: $B \longrightarrow I \rightarrow F$ $_m$				
	⫫Gauss	⫫Cop.	Gauss	Cop.	KDE
Bag Acc.	**98.2**%	**96.9**%	**97.1**%	**95.7**%	88.6%
Log Lik.	-38035	-38209	-37980	-38143	-35999

variance, leaving us with $p = 76$ features. Results are not state-of-the-art, but moderately good; among our models the BIF model with independent Gaussians for $P(F|I)$ has the highest cross-validation bag accuracy and log-likelihood. In addition to the assumptions, one main difference between EMG muscle datasets and the MUSK1 dataset is the number of features. There is a need for a better way of capturing the dependence relationships among this (and larger for other problems) number of features. By "better", we mean more compact but still tractable. In Chapter 6, sum-product networks are shown to provide this compact and tractable representation, which leads to higher cross-validation bag accuracy and log-likelihood on the MUSK1 dataset.

TABLE 5.3: Results of the MUSK1 dataset.

MUSK1	BIF: $B \multimap\!\!\boxed{\rightarrow I \rightarrow F}_m$					FIB: $B \leftarrow\!\!\boxed{I \leftarrow F}_m$			
	$\perp\!\!\!\perp$Gauss	$\perp\!\!\!\perp$Cop.	Gauss	Cop.	KDE	LR	KNN	SVM	QDA
Bag Acc.	**86.5**%	**84.4**%	69.6%	.641%	**77.2**%	**78.3**%	**77.2**%	**83.2**%	**83.1**%
Log Lik.	*-14921*	-18437	-45815	-51031	-33591	-34086	-34120	*-34001*	-34056

5.5.4 Comparison with Weakly Supervised Learning

As per bag accuracy results (first two rows are taken from Tables 4.2 and 5.1) are shown in Table 5.4. The similarity graph model that ultimately led to the highest bag accuracy; Probabilistic threshold minimum graph, is in use for the weakly supervised learning paradigm. The model used is the BIF model where $P(F|I)$ is estimated using Gaussian independent features. Results of the first two rows show that this BIF model leads to a better bag accuracy in both Upper Leg and Lower Leg datasets. The last two rows show the results of collecting all instances of the same bag label in one bag. In this case, the spectral graph-theoretic grouping based weakly supervised learning achieves higher accuracy. When number of bags is small, this is not a problem for the spectral graph-theoretic grouping phase because spectral grouping collects all unlabelled instances of the same bag label in one graph (as if it is one bag) before starting to group them on the graph. However, generative MIL models need some population of bags so that they can generalise and assign probabilities involving B on some basis. For example, assume we have two bags in total labelled 1, one containing 500 instances labelled 1 and 500 instances labelled 2 and the other containing 10 instances labelled 1 and 0 instances labelled 2. The value of $P(I|B)$ in the case when they are two bags is very different from its value with one bag containing 510 instances labelled 1 and 500 instances labelled 2.

As mentioned in Chapter 4, the generative MIL models are more flexible in handling different assumptions as per the bag-of-instances setting of data. This is why the MUSK data is not tested on the weakly supervised learning paradigm but it is tested here with MIL. Again, another advantage of generative MIL models is their capability of generating simulated data as the joint distribution of data and label is modelled.

TABLE 5.4: Comparison between the generative MIL model BIF and weakly supervised learning paradigm on the EMG datasets.

	⫫Gauss BIF	Prob. threshold Min Weak Class.
Upper Leg	95.5%	95%
Lower Leg	98.6%	96.1%
Upper Leg (instances of same bag label in one bag)	89.1%	95%
Lower Leg (instances of same bag label in one bag)	91.3%	96.1%

5.6 Ad hoc Measures for EMG

An ad hoc measure of confidence in the characterisation of a muscle and another measure of level of involvement (LOI) of a certain MUPT into a certain disorder are provided in this section. These ad hoc measures are of use only to EMG muscle data as they lead to sound experimental results. These measures are not intended to be of use to any other data.

5.6.1 Measure of Confidence

A confidence measure reports the uncertainty of a certain muscle characterisation, which is the uncertainty of the classification of a muscle. In other words, it measures the degree to which an output of a muscle classifier is likely to be correct.
One advantage of the generative models presented is that they provide probabilities of each bag label assignment. Therefore, in the BIF structure, the value $P(B = b|\vec{F} = \vec{f})$ for the muscle label, i.e. the label that leads to highest value of the conditional probability, is taken as a representation of the confidence in the muscle characterisation.

5.6.2 Measure of Level of Involvement (LOI)

Level of involvement (LOI) measures the level of severity of a certain disorder for each motor unit potential train (MUPT). Using the BIF structure, for each instance I, the 3 values (one per label) of the conditional probability $P(I = i|\vec{F} = \vec{f})$ are calculated for each I. This is the conditional probability of I given F marginalised over B.

These values were used in an exploratory analysis showing how the change of values of a single feature or two features together affects the LOI values of MUPTs on average. The following graphs show how the average values of LOI over all the MUPTs of all the muscles change along with changes in feature values.

Figures 7.1-7.4 show the average LOI values along with the corresponding values of 4 individual features; turns, amplitude, area and thickness, respectively.

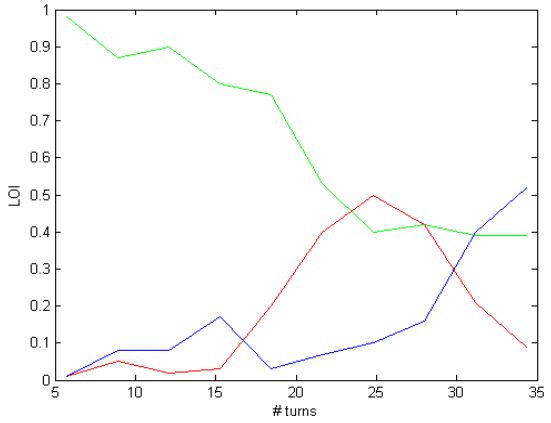


FIGURE 5.2: Average LOI values of all MUPTs vs. change of values in the "turns" feature. Normal LOI is displayed in green, myopathic LOI is displayed in red and neurogenic LOI is displayed in blue

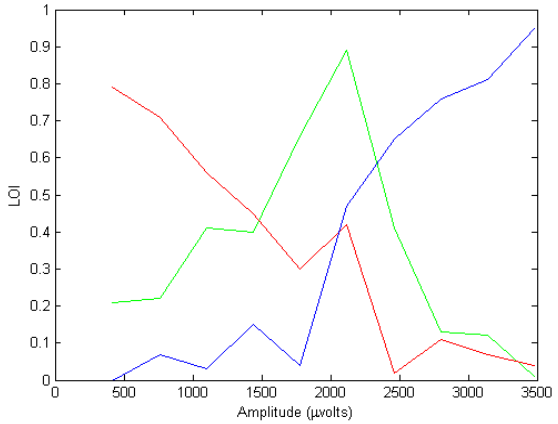


FIGURE 5.3: Average LOI values of all MUPTs vs. change of values in the "amplitude" feature. Normal LOI is displayed in green, myopathic LOI is displayed in red and neurogenic LOI is displayed in blue

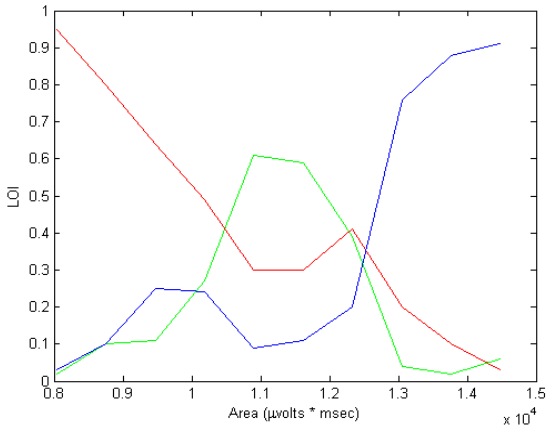


FIGURE 5.4: Average LOI values of all MUPTs vs. change of values in the "area" feature. Normal LOI is displayed in green, myopathic LOI is displayed in red and neurogenic LOI is displayed in blue

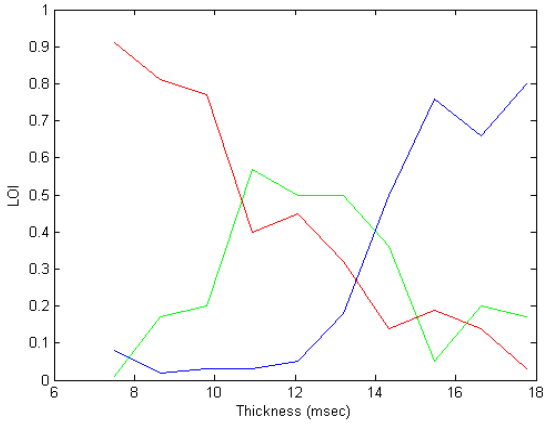


FIGURE 5.5: Average LOI values of all MUPTs vs. change of values in the "thickness" feature. Normal LOI is displayed in green, myopathic LOI is displayed in red and neurogenic LOI is displayed in blue

Figure 7.5 shows the average LOI values along with the corresponding values of two features; turns and amplitude.

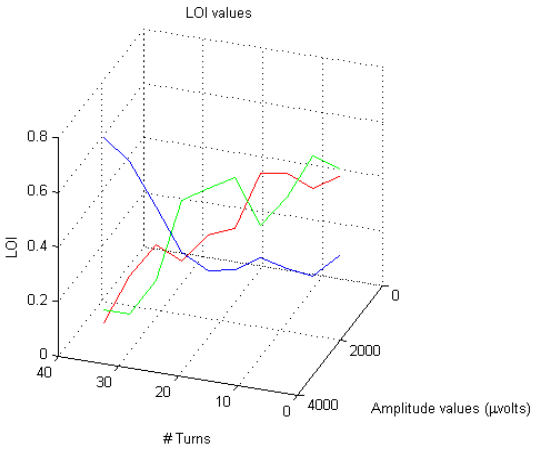


FIGURE 5.6: Average LOI values of all MUPTs vs. change of values in "turns" and "amplitude" features. Normal LOI is displayed in green, myopathic LOI is displayed in red and neurogenic LOI is displayed in blue

Figure 7.6 shows the average LOI values along with the corresponding values of two features; amplitude and area.

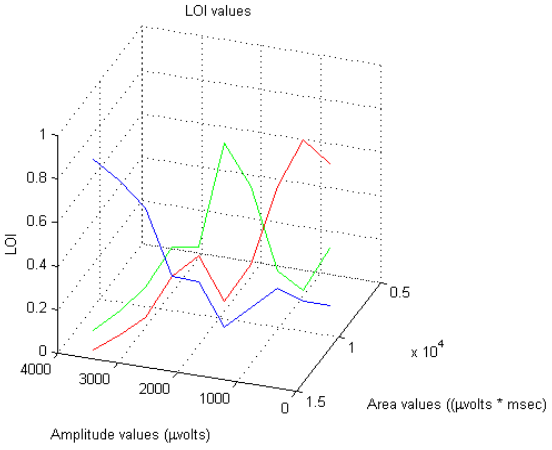


FIGURE 5.7: Average LOI values of all MUPTs vs. change of values in "amplitude" and "area" features. Normal LOI is displayed in green, myopathic LOI is displayed in red and neurogenic LOI is displayed in blue

Figure 7.7 shows the average LOI values along with the corresponding values of two features; area and thickness.

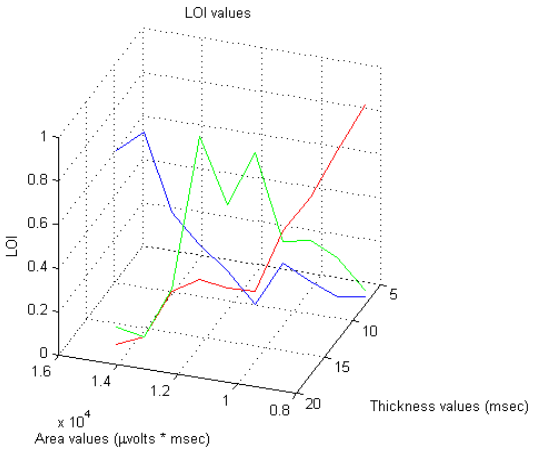


FIGURE 5.8: Average LOI values of all MUPTs vs. change of values in "area" and "thickness" features. Normal LOI is displayed in green, myopathic LOI is displayed in red and neurogenic LOI is displayed in blue

5.7 Summary

Results on the EMG data indicate that all of the BIF-based generative models perform better than previous state-of-the-art results of muscle classification. This demonstrates that modelling the EMG as a two-stage generative model according to the way these signals are actually generated in real life significantly improves prediction accuracy over previous strategies on this task on the bag level accuracy. As far as muscle classification accuracy is concerned, the parametric model components (Gaussian and Copula) appear best.

Also, a general paradigm for generative models in Multiple-Instance Learning (MIL) has been introduced. This paradigm may be useful in modelling other MIL tasks in addition to the EMG muscle datasets, and it was tested on the MUSK1 dataset. Although MIL is a well-developed sub-field of Machine Learning, before the work of Adel et al. in [Adel et al., 2013] where the results shown in Table 5.1 and Table 5.3 have been published, generative modelling approaches had not received much attention. The results suggest that models that are well aligned with the actual data generation in a problem domain (the BIF structure in the case of the EMG signal classification task) can be a good choice for classification and modelling purposes.A comparison with the weakly supervised learning paradigm is also provided.

Regarding the MUSK dataset, bag accuracy is far from state-of-the-art accuracy on the same data (which is 96%) [Tao, 2004]. It is believed that one of the reasons is the fact that the MUSK dataset has many more features than EMG (168 vs. 8) and if the joint feature conditional probability distribution $P(F|I)$ can be represented more efficiently, bag accuracy can improve. Chapter 6 discusses in more detail using sum-product networks (SPNs) to provide an efficient representation of features.

Chapter 6

Sum-Product Networks (SPNs)

6.1 Motivation

As was the case with the MUSK dataset in Chapter 5, when number of features is not small, and especially when it becomes large, the need for a compact and tractable representation of the dependence relationships among features grow. In this part of the thesis, we are mainly interested in constructing a model that can represent joint feature probability distributions efficiently. The need for such representation was not crucial with EMG datasets as they have far too few features, but the MUSK dataset is an example of a multiple-instance learning (MIL) dataset with 168 features. In this age of Big Data, dimension of a dataset can be even much more than 168. Therefore, the need for graphical models that can provide an efficient representation of complex probability distributions becomes crucial. This is the motive behind exploiting sum-product networks (SPNs). SPNs are mainly used here to model $P(F|I)$ in the BIF generative model presented in Chapter 5 as an MIL paradigm. However, SPNs can be used to model probability distributions in numerous other problems and applications.

Note that, in this chapter, generically using the word "model" refers to a representation of a joint variable probability distribution. For example if we note that Bayes nets do not efficiently represent a model, we do not refer to a specific MIL generative model of Chapter 5 but to a generic joint variable probability distribution model.

Bayes nets can compactly represent probability distributions but they are not guaranteed to provide an efficient representation. For example, Bayes nets may turn inference into an intractable process, making the use of approximate inference inevitable. On the other hand, the main edge of SPNs is their capability of representing a wide range of probability distributions, with their respective conditional dependence complications, in a compact and tractable model. The structure of an SPN needs to be learnt unless a fixed structure is used. The difference between one SPN structure and another can lead to differences in the model's performance, which in our MIL case is measured by bag accuracy.

In this chapter, we present a sum-product network (SPN) structure learning algorithm via applying biclustering motivated by an algorithm called rank-one downdate (R1D). The main premise of the introduced algorithm is to build the structure learning process upon finding subSPNs that are the most coherent without having to pick the most fitting subSPN available via one separate variable or instance splitting at each step. Results show that SPNs learnt by the introduced algorithm, and used to model $P(F|I)$ in the BIF structure presented in Chapter 5, ultimately lead to higher bag accuracy than those obtained by density estimators and than those obtained by other SPN structures, on the MUSK dataset. The introduced SPN structure learning algorithm is also significantly faster than its counterpart algorithms in terms of SPN structure learning time. The bulk of the content of this chapter as well as a more comprehensive set of results are presented in Adel and Ghodsi [2014].

6.2 Graphical Models and Sum-Product Networks (SPNs)

Probabilistic graphical models are graphs that express the conditional dependence relationships among random variables of a joint distribution. Graphical models have applications in computer vision, speech recognition, medical diagnosis and many other fields. One of the main advantages of graphical models is their ability to compactly represent complex distributions [Roth, 1996]. Two main types of graphical models are Bayesian networks and Markov random fields. Bayesian networks model asymmetric dependencies as they depict each variable as a node and depict each conditional dependence relationship as an arrow.

On the other hand, Markov random fields model symmetric dependencies as each variable is represented by a node and each symmetric relationship is represented by an edge [Koller and Friedman, 2009]. The number of parameters in a graphical model is typically an exponential function of the number of random variables. Therefore, inference can be intractable in graphical models due to the complex interactions between random variables and hence approximate inference is required.

This highlights one of the main uses of latent variables in graphical models. Latent variables can be used to reduce the number of parents of nodes in a model while still allowing the related probability distribution to be represented. Deep architectures can be considered a type of a graphical model, which is based on introducing multiple layers of latent variables [Hinton et al., 2006]. Using deep architectures, problems containing complex interactions between variables, which were difficult to model compactly using a Bayesian network, can be modelled using a rich yet compact deep architecture [Bengio, 2009]. However, inference is not guaranteed to be efficient in deep architectures. In other words, deep architectures represent a step forward compared to Bayesian networks and Markov random fields in terms of compactness of the model created but not in terms of tractability of inference. Therefore, inference is intractable in deep architectures and approximate inference is still required. Approximate inference has many problems; most notably, it is unreliable and there is no guarantee it will lead to good results.

Thus, the main requirements for a model to be considered a good representation of a complex joint probability distribution are compactness and inference efficiency. The above representations, especially deep architectures, manage to accomplish the first but not the second requirement. This is why sum-product networks recently came into the literature. Sum-product networks (SPNs), introduced by Poon and Domingos [2011], aim at reaching the most compact representation possible provided that inference remains tractable. Layers of hidden variables are added to the model as long as they increase compactness and as long as they keep inference tractable. Thus, SPNs guarantee that the resulting model is rich as well as tractable.

An SPN can be defined as either a univariate distribution, weighted sum of other SPNs each

containing the same set of variables, or a product of other SPNs with disjoint sets of variables. The "scope" of an SPN is the set of variables that appear in the leaf nodes of the SPN, or in other words, the set of variables such SPN contains. A tractable univariate distribution is an SPN [1]. This univariate distribution can be any type of distribution; e.g. Bernoulli (representing boolean random variable), multinomial (discrete random variable), Poisson (continuous random variable) or any other individual distribution as long as it is tractable. SPNs provide a representation in which these single tractable distributions are combined into a richer and more complex distribution provided that the resulting distribution is tractable. This can be achieved by applying two combination rules:

- A product of SPNs over disjoint variables is an SPN. An SPN with this property is referred to as a decomposable SPN. The meaning of this property is related to the independence among variables; for instance a product $X_1 \times X_2$ means that X_1 and X_2 are independent. In the original definition provided by Poon and Domingos [2011], they allow for a less restrictive version of this condition referred to as consistency of an SPN as they note that the scopes do not have to be disjoint but it is enough if one variable does not appear negated in one child of the product node and non-negated in another. However, we do not need this less restrictive version in the work in this chapter.

- A weighted sum of SPNs that have the same scope is an SPN. Weights must be positive [Gens and Domingos, 2013]. An SPN with this property is referred to as a complete SPN. This property depicts mixture models where each subcomponent of an SPN (subSPN) of the weighted sum acts as a mixture component. A sum of SPNs can be thought of as the result of summing out a hidden variable.

An SPN can then be defined as a rooted directed acyclic graph (DAG) whose leaves are univariate distributions, and whose internal nodes are sum and product nodes. As noted earlier, every edge from a sum node to one of its children has an assigned positive weight.

[1] A tractable univariate distribution is defined as a univariate distribution whose partition function and mode can be computed in time $O(1)$ [Gens and Domingos, 2013].

In the case of Bayes nets, which entirely depend on conditional independence, if there is no conditional independence among the variables of a model, then the resulting net consists of a big clique. If an SPN is used instead, the resulting representation can well be compact because, unlike typical Bayes nets, an SPN has the advantage of utilising context-specific independencies. Context-specific independencies are independencies that hold in some contexts but not in others [Boutilier et al., 1996]. In other words, they are independencies that hold given a specific assignment of values to specific variables [Boutilier et al., 1996]. In Bayes nets, inference can be exponential in the number of variables. On the other hand, in SPNs, inference is always linear in the size of the network. The size of the network can be exponential in the number of variables but much more often than not this is not the case and consequently there are many complex problems that can be represented compactly and efficiently using SPNs but not using typical Bayes nets.

6.2.1 SPNs as a Part of the BIF Generative Model

As an example of an SPN, let's assume we have two variables (for simplicity) F_1 and F_2. Each of the two variables is a binary variable. Let's assume that an SPN is required to represent the joint probability distribution of F_1F_2. One example of a resulting SPN is displayed in Figure 6.1. In this example the two variables are not independent but they are context-independent, which would have made it a clique if a Bayes net was used. This means that if specific mixtures over these two variables can be found so that variables are independent given their specific joint assignment of values in every mixture, we can add a hidden variable to express the mixture by a sum node (the sum node at the top of the network in Figure 6.1) and then express context-specific independence in each mixture by a product node (the two product nodes on the second layer of the network). The third layer contains sum nodes representing the univariate distribution of each variable in each mixture. There are two values only of each variable as F_1 and F_2 are binary. Note that subSPNs are reused in larger components, e.g. the subSPN represented by a sum node at the right side of third layer, which increases compactness of the network.

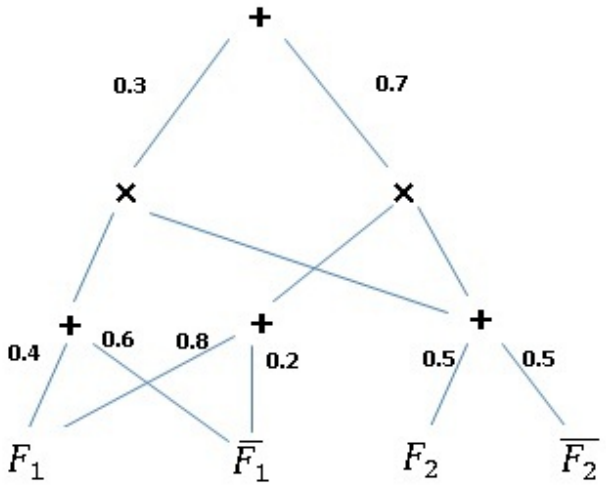


FIGURE 6.1: An SPN representing a model with two variables.

The last paragraph illustrates one simple example of an SPN. Now we want to show how an SPN can be used in the BIF model structure of MIL. Let's assume that the two variables in Figure 6.1 are two instance features in the BIF model. Let's assume that an SPN is is required to represent the probability distribution $P(F|I)$ where $t = 2$, i.e. $I \in \{1, 2\}$. As we want to keep the arrow from I to F in order to maintain the advantages of the BIF structure, we can model the joint $P(F_1F_2|I)$, not $P(F_1F_2I)$. In case F_1 and F_2 are independent given I, then Figure 6.1 is a possible representation of $P(F|I)$ with the sum node at the top representing $I = 1$ with probability 0.3 and $I = 2$ with probability 0.7. In case F_1 and F_2 are not independent given I, then $P(F|I = 1)$ on its own can be assumed to have a representation similar to the one in Figure 6.1, and the same goes for $P(F|I = 2)$ while both are linked with a sum node at a layer above the layer at the top of Figure 6.1, which represents the mixture over $P(F|I)$. Put another way, in case F_1 and F_2 are independent given I, I can be considered the hidden variable used in specifying the context of the variables where they are independent. However, if F_1 and F_2 are not independent given I, then one hidden variable is needed for $P(F|I = 1)$ and another for $P(F|I = 2)$. Even if the latter case further demonstrates the advantage of SPN, let's assume the former holds. The assumption that the former holds is valid not only for the sake of simplicity but also because two consecutive sum nodes can be mixed in one, so let's assume that the former holds which makes Figure 6.1

a possible representation of $P(F|I)$. Assume $I = 1$ happens with a probability 0.3 and is represented by the subSPN at the left side of Figure 6.1, whereas $I = 2$ happens with a probability 0.7 and is represented by the subSPN at the right side of the figure.

In the above example, features are assumed to be binary. However, this is not the case in the MUSK dataset as their features are continuous. We assume that the univariate feature distributions are Gaussian. Therefore, the leaves of the SPN used to represent the joint feature distribution of the MUSK dataset represent univariate Gaussian distributions, rather than binary distributions as in Figure 6.1.

Regarding the learning phase of the BIF structure, hard EM is used and the steps are shown in Algorithm 1 in Chapter 5. In this and the next paragraph, the same learning steps are explained with the difference being in using SPNs, rather than the density estimation techniques mentioned in Chapter 5, to model $P(F|I)$. Before the first iteration, I has a value from the instance initialisation step. Next is the parameter estimation step, presented in Section 5.4.1.1 (M-step), for every iteration of EM, we have both I and $\vec{F}$ observed and we wish to model the distributions $P(\vec{F}|I = i)$ for $i = 1, 2, ..., t$. Assuming $t = 2$, two subSPN structures should be learnt at every iteration; one for $P(\vec{F}|I = 1)$ and another for $P(\vec{F}|I = 2)$. This demonstrates how important it is to have a fast SPN learning algorithm as when the number of features is large, a slow SPN structure learning algorithm can slow down the whole learning process. SPN structure learning is what this chapter mainly introduces, therefore the algorithm assigned for this task will be presented in detail throughout the chapter.

Back to Hard EM. The next step is the label updating step, presented in Section 5.4.1.2 (E-step) where the probability $P(\vec{F} = \vec{f}|I = i)$ for $i = 1, 2, ..., t$ should be computed. The same probability is required in inference, presented in Section 5.4.2. Recall that F is observed. The rest of this paragraph is dedicated to show how to compute the probability of query variables given evidence variables on an already constructed SPN. This represents

how $P(\vec{F} = \vec{f}|I = i)$ is computed in SPNs. One of the quantities that are always tractable in SPNs because they can be computed in linear time in the size of the network is the conditional probabilities. Conditional probabilities are evaluated as the ratio of the two respective marginals. In order to compute the $P(\vec{F} = \vec{f}|I = i)$, two marginals are computed; $P(\vec{F} = \vec{f}, I = i)$ and $P(I = i)$. The former is then divided by the latter so that a value of $P(\vec{F} = \vec{f}|I = i)$ is obtained. Marginals can be computed by summing out one or more variables. All leaf nodes belonging to such variables are set to 1. In order to start computing a marginal, the value at each leaf node is computed first. The value at a leaf node represents a variable value. For example, if the observed binary $\vec{F} = (1, 0)$, then $F_1 = 1$, $\bar{F}_1 = 0$, $F_2 = 0$ and $\bar{F}_2 = 1$. The weighted sum of the child nodes of a sum node represents the value assigned to a sum node and the product of the child nodes of a product node represents the value assigned to the product node. Leaf nodes of variables not identified in the marginals are the leaf nodes that are all set to 1 as they represent the variables to be summed out or marginalised over. Evaluation of the SPN nodes continues in this fashion until the root is reached. For example, assume again that the subSPN at the left side of Figure 6.1 represents $I = 1$ and the right side represents $I = 2$, then $P(\vec{F} = (1, 0)|I = 1)$ can be calculated as:

$$P(\vec{F} = (1, 0)|I = 1) = \frac{P(\vec{F} = (1, 0), I = 1)}{P(I = 1)} = \frac{0.4 \times 0.5 \times 0.3}{0.3} = 0.2$$

6.3 Rank-One Downdate (R1D) Algorithm

Rank-one downdate (R1D) algorithm was developed by Biggs et al. [2008] to compute a nonnegative matrix factorization (NMF) of matrices with nonnegative entries. By matrix, we refer to a data matrix consisting of instances and variables. The goal of NMF is to approximate such matrices by a product of two low-rank matrices W and H whereas each of them contains only nonnegative entries. The algorithm is partly based on the singular value decomposition (SVD) as it computes the dominant singular vectors and values of submatrices that are dynamically determined by the algorithm itself [Biggs et al., 2008].

A rank-one (R1) submatrix is extracted on each iteration from the original data matrix according to an objective function. Our SPN structure learning algorithm is adapted from R1D but it is not the same. One of the reasons R1D was chosen as a basis for our algorithm is the fact that variable splitting can be turned into correlation splitting while instance splitting remains based on the similarity between instance vectors. Another reason is the speed of the algorithm compared to its counterpart NMF algorithms and this is the case for our adapted SPN structure learning version as well.

According to the Perron-Frobenius theorem [Golub and Loan, 1996], the leading singular vectors of a nonnegative matrix have nonnegative values. Consequently, computing an R1 NMF of a nonnegative matrix is a trivial task [Biggs et al., 2008]. R1D extends this idea to compute higher order NMF by taking a submatrix of the original data matrix, computing an R1 NMF of this submatrix and subtracting the R1 NMF from the original submatrix while setting all negative values of the resulting submatrix to 0. The algorithm proceeds iteratively whereas each iteration corresponds to computing one higher rank. One reason why this algorithm is a better candidate for clustering than the original SVD is that each iteration acts on a submatrix rather than the whole original matrix. Taking the whole matrix has the side effect of averaging the variables/instances while taking a submatrix makes clustering and biclustering more straightforward. In the case of biclustering, variables and instances corresponding to the nonzero singular vector positions are considered a bicluster. Another advantage of extracting a submatrix rather than working on the full matrix, which is fundamental for our cause, lies in the fact that a correctly chosen submatrix is closer or more likely to be of rank one (remember that this is an approximation) than the full matrix [Biggs et al., 2008].

As such, the R1D algorithm mainly consists of an outer loop and an inner loop. Each iteration of the outer loop, which is referred to by Biggs et al. [2008] as "greedy R1 downdating", has as an output one column of W as well as one column of H. It does so by iteratively extracting the best possible candidate for an R1 submatrix of the data matrix (task of the inner loop) and subtracting it from the original matrix. We care more about the inner loop here as we are not interested in calculating an NMF of a matrix. Assume that the original

matrix is $A \in R^{m1 \times n1}$. The inner loop extracts an R1 submatrix via computing 5 values; $M1$, $N1$, $u1$, $v1$ and $\sigma 1$. $M1$ is a subset of $\{1, 2, ..., m1\}$, $N1$ is a subset of $\{1, 2, ..., n1\}$, $u1 \in R^{m1}$, $v1 \in R^{n1}$ and $\sigma 1 \in R$ whereas both $u1$ and $v1$ are unit vectors. The goal of the inner loop is to select these 5 values so that the submatrix indexed by rows $M1$ and columns $N1$ is approximately (as close as possible to be) of rank one. The objective function used in order to achieve this goal is as follows [Biggs et al., 2008]:

$$f(M1, N1, u1, \sigma 1, v1) = \|A(M1, N1)\|_F^2 - \gamma \|A(M1, N1) - u1\ \sigma 1\ v1^T\|_F^2 \qquad (6.1)$$

The γ parameter is a penalty parameter that leads to the objective that the farther $A(M1, N1)$ departs from being of rank one, the larger the penalty.

6.4 Related Work

Structure learning in SPNs has not been a hot topic in the SPN literature until recently. In the beginning, only parameters were learnt. A structure was readily given and weights were learnt either generatively [Poon and Domingos, 2011] or discriminatively [Gens and Domingos, 2012]. Hard EM algorithm was used by Poon and Domingos [2011] to apply generative parameter learning on SPNs. Deep SPNs were learnt successfully using hard EM, but a pre-defined network structure was used. The same persisted in the discriminative parameter learning algorithm introduced by Gens and Domingos [2012], which was based on a gradient descent algorithm.

The first algorithm that learnt the structure of an SPN from data was proposed by Dennis and Ventura [2012]. After an initial step where instances are clustered and a sum node is created accordingly, they start to cluster variables in a top-down approach creating product nodes. As they do not cluster instances after the first step, context-specific independences that appear from this point onwards are not taken into consideration. Such context-specific independences represent the main reason why SPNs can be preferred over Bayesian networks as SPNs accomplish better likelihood values due to these independences and, consequently,

ignoring them penalises the whole expressive power and richness of the model. Also, clustering ignores correlation between variables and as a result two variables that are strongly correlated but do not have similar values will be clustered in two different clusters, which would potentially lead to suboptimal likelihood values. In addition, weights related to sum nodes are learnt after, rather than during, the structure learning process. Peharz et al. [2013] introduced another SPN structure learning algorithm based on greedily merging small image regions into larger regions in a bottom-up approach. An algorithm for online SPN structure learning was proposed by Lee et al. [2013], where the problem is cast into an online clustering problem. They develop an incremental structure learning algorithm based on dynamically modifying the number of clusters, which in turn is based on the incoming data. The most eminent SPN structure learning algorithm so far was proposed by Gens and Domingos [2013]. They apply a recursive top-down approach where, at each step, variables are checked if they can be split into approximately independent subsets in which case a corresponding product node is the outcome. Otherwise the current set of instances is clustered and a sum node is returned whose weights are calculated by the proportions of instances belonging to every cluster. This algorithm greedily optimises log-likelihood and overcomes several limitations that existed in Dennis and Ventura [2012]. However, it keeps searching locally, not globally, at the ideal candidate for a splitting at each step. The matrix in Table 6.1 shows an example where the best decisions for variable/instance splittings, and consequently the log-likelihood value, are conditioned by the local search nature of the algorithm. We will return to this example later. Also, deciding which subsets of variables are independent of one another is based on approximations that do not always lead to accurate variable clusters. Granted, searching for independence always involves such risk but accumulating these approximations step after step can lead to inaccurate subSPNs. The algorithm is applicable on discrete as well as continuous data but the reported results were on binary data only. Finally the outcome of the algorithm is a tree SPN and subSPNs are not reused. The last algorithm, to the best of our knowledge, that dealt with SPN structure learning was proposed by Rooshenas and Lowd [2014] where the authors adapt an SPN structure learning methodology based on both mixture modelling and arithmetic circuits (AC) learning. It is different from the other

algorithms in the sense that it does not only apply local modifications in the search of an optimal model as AC learning "could" lead to global changes but a global look at the data is not guaranteed in each step.

6.5 SPN Structure Learning Algorithm (SPN-R1DBiclus)

We introduce an SPN structure learning algorithm that is based on biclustering. The algorithm aims at hierarchically finding subSPNs by applying biclustering to the data matrix and ultimately converting the biclusters obtained into an SPN structure. Biclustering is the problem of simultaneously clustering rows and columns of a data matrix. The primary premise of the algorithm is to avoid being restricted by having to split the data matrix at the rate of one dimension split at a time because this could potentially result in splitting elements that should have belonged to the same subSPN. This drawback can happen because local inspection applied on the data matrix before each splitting did not discover this subSPN as it was not only one sum node away, nor only one product node away, but at least (nearest) as far as one sum and one product node away. In other words, when it is impossible to reach the most coherent subSPN as both sum and product nodes should be applied concurrently at the same step, and when applying one sum or product node makes this subSPN already divided at a previous step, this can lead to lower log-likelihood values, and this is one of the main issues that our algorithm aims at handling. We refer to the introduced algorithm as SPN-R1DBiclus. The algorithm is presented in Adel and Ghodsi [2014].

The biclustering algorithm we introduce to build SPNs is based on the rank-one downdate (R1D) algorithm developed by Biggs et al. [2008]. We do not do downdating here and the word "downdate" is kept in reference to the original name "R1D" only.

A toy example is provided in Tables 6.1 & 6.2 in order to compare between the steps taken by the algorithm proposed by Gens and Domingos [2013] (SPN-Gens) and our introduced algorithm (SPN-R1DBiclus), respectively, as well as the log-likelihood values obtained in each case. Table 6.1(A) shows the original data matrix which consists of 6 instances with

TABLE 6.1: SPN-Gens on an Example Data Matrix

40	40	40	40
40	40	40	40
40	18	18	40
20	18	18	20
20	20	20	20
20	20	20	20

A

40	40	40	40
40	40	40	40
40	18	18	40
20	18	18	20
20	20	20	20
20	20	20	20

B

40	40	40	40
40	40	40	40
40	18	18	40
20	18	18	20
20	20	20	20
20	20	20	20

C

40	40	40	40
40	40	40	40
40	18	18	40
20	18	18	20
20	20	20	20
20	20	20	20

D

4 variables each, whereas each horizontal line represents one instance. SPN-Gens starts by applying clustering in the first step and the result in this case is placing the first 3 instances together in one cluster and the other 3 instances in another, as shown in Table 6.1(B). In this step, the cluster of elements with value 18 is split. The following step in SPN-Gens is related to the top cluster as the independence check is applied first and it results in splitting the first variable from the others as shown in Table 6.1(C). The first and fourth variable had the same values across all instances but the first variable is still rated as a variable that is "approximately" independent from the other group of variables containing the second, third and fourth variable. The same happens in step D. Variables keep getting split until we reach multinomial variables, which represent the leaves of the SPN. The problems in steps C and D are related to the mutual information dependence check in SPN-Gens. We focus more on step B as it is related to the main idea of the algorithm rather than the subroutine chosen to check independence. The point here is that the elements with value 18 could have been a perfect and coherent chunk on which a subSPN could be based but due to the fact that these elements are split from one another in a previous "local" splitting (step B), the algorithm is not able to put them in one SPN. The final log-likelihood value on the same data is -2.

On the other hand, Table 6.2 shows how SPN-R1DBiclus deals with the same data matrix of Table 6.1(A). By applying biclustering, the algorithm spots the chunk of elements with value 18 from the beginning and places them in one bicluster. In order to adjust biclustering so that it can cope with how SPNs are made, we need to get parts of the data that share aligned instances (rows) and variables (columns) so that they can be aligned as a submatrix and can be a part of the resulting SPN. Elements that do not belong to the "18" bicluster need to be divided as either in Table 6.2(B) or, reversely, by splitting columns 2 and 3 from the other two then clustering columns 2 and 3 into 2 clusters; the 18 cluster and another. Splitting as in Table 6.2(B) is chosen here because the other option assumes that variables 2 & 3 are always independent even in cases of instances not belonging to the "18" bicluster, which is not true. Table 6.2(B) does not adopt this assumption as it merely clusters instances that belong to the bicluster; i.e. their variables that do not belong to the bicluster, in a cluster. To put it in simpler terms, we follow a more "conservative' approach, regarding variable independence, that does not consider two groups of variables to be independent unless we are confident they are at least uncorrelated across all checked instances. Figure 6.2 shows the SPN constructed via Table 6.2. Rows represent instances and columns represent variables. Now we have 3 chunks of data resulting from one iteration of R1D biclustering; one is the bicluster chosen by the algorithm, another consists of variables, which belong to instances of the bicluster but do not belong to the 18 bicluster. A third cluster is the one consisting of instances not belonging to the bicluster. Weights of edges of a sum node are assigned based on the proportion of elements in each cluster. In our example, weight of the left side cluster is 0.33 and weight of the right side cluster is 0.66. Note that this ultimately does not always lead to 2 clusters for the whole matrix because the algorithm continues acting recursively in each of the three resulting paths. For example the right side cluster of the displayed sum node can get clustered again and this will be equivalent to having more than 2 clusters for the whole displayed matrix. The same applies to variable groups and product nodes. The final log-likelihood value in this case is -1.39, which is better than the result of Table 6.1. As SPN-R1DBiclus proceeds recursively through each of the three parts. In case the bicluster returns itself again, this means that the bicluster is the best approximate

rank-one (R1) submatrix possible to be returned from the algorithm and a *multivariate leaf node* is reached. Being an R1 matrix means that variables are all dependent and we can proceed with one big sum node (each instance is a branch with a weight = 1/#instances) followed by one product node for variables of each instance. Back to our example, the 18 bicluster does not have any independent variable subgroups (it is R1) and this is why a sum node followed by a layer of product nodes not only speeds up the recursive process but also improves log-likelihood as it does not assume any nonexistent independences.

TABLE 6.2: SPN-R1DBiclus on the Same Data Matrix

40	40	40	40
40	40	40	40
40	18	18	40
20	18	18	20
20	20	20	20
20	20	20	20

A

40	40	40	40
40	40	40	40
40	18	18	40
20	18	18	20
20	20	20	20
20	20	20	20

B

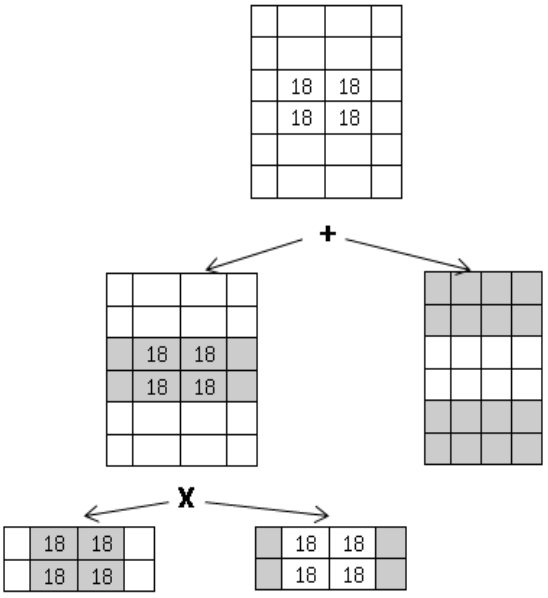


FIGURE 6.2: Status of the SPN after Table 6.2 (B)

The objective function used in nearly all SPN learning algorithms till now; which is the likelihood function, is different from the parameter estimation objective function of SPN-R1DBiclus; which is the least squares. It has been stated by Hofmann [1999] and others

that the likelihood function has advantages over the least squares fit in data analysis and parameter estimation. The likelihood function proved beneficial in several applications, and, in general, it has more advantages than drawbacks over the least squares fit. As SPNs are graphical models, it was natural that the likelihood function is chosen as the objective function for SPN learning. However, SPNs split the data in two, rather than one, directions. To the best of our knowledge, all SPN learning algorithms in the literature either split the data in one direction at a time, or focus on one direction (variables) throughout the whole algorithm. Moreover, there is no one likelihood objective function that controls data splitting in both directions operated upon (both functions are usually likelihood-based but not the same objective function is optimised in both the variables and instances directions). Our introduced SPN learning algorithm is based on the fact that the least squares function used has the advantage of acting upon both splitting directions (variables and instances) concurrently. Moreover, the version we apply, which is partly motivated by computing the SVD, avoids some of the inherent drawbacks of most similar least squares fit-based methods. It is worth noting that the use of these singular values here is merely to pick out some variable/instance indices or, in other words, to cluster the data matrix. This means that no singular values are carried through the algorithm as the goal of this singular values is to pick out certain indices. This also helps avoid further drawbacks of SVD methods as we do not care about the values themselves but use them for matrix splitting.

In SPN-R1DBiclus, note that rows represent instances and columns represent variables. One main subroutine; $R1Biclus$, of the algorithm extracts a bicluster, which is a subset of variables and instances. The bicluster extraction represents the part mostly motivated by the R1D algorithm. A resulting bicluster is an approximate rank-one submatrix. In case the submatrix returned is equal to the full subroutine input matrix B, this means that the input matrix B is already of rank-one, or more precisely as close as it can get by the subroutine to be of rank-one. In this case a boolean called *stop* is set to true. Along with each bicluster extraction, the rest of the matrix is split into two, as shown in Table 6.2(B), so that the three resulting parts; the bicluster and the two other parts of the matrix, form a sum node on the whole matrix then a product node on one of the branches (clusters) of the sum node.

Weights of a sum node are assigned by the proportions of instances belonging to each cluster. The algorithm proceeds recursively on the three parts until a base case is reached. There are three base cases here. The first is when the current matrix has one variable only (*line* 6 in Algorithm 2). In this case the remaining vector represents a univariate distribution and a corresponding leaf node is created. The second base case is when the current matrix has one instance only (*line* 9). In this case, all variables can be considered independent and a product node is created along with its leaves being the involved variables. Before getting back to the third base case which is related to the boolean $stop = true$, let's move to when $stop = false$; which means that the subroutine $R1Biclus$ returns a submatrix that does not span the whole subroutine's input matrix B. In this case, the bicluster MB corresponds to a subSPN on which the subroutine will recursively run. Now regarding the rest of B, back to Table 6.2, we are now standing at Table 6.2(A) where MB is the "18" square. We now have two parts of B; which are MB and $B \setminus MB$. As our aim is to learn an SPN, we need to align this couple of submatrices so that we can continue building the SPN. This makes it a must to decide one of two ways regarding how to split that part $B \setminus MB$. Is it by splitting all variables into two groups first; one representing variables belonging to MB and one for the rest, or by splitting all instances into two groups first; one representing MB's instances and one for the rest as in Table 6.2(B)? The former option assumes that MB's variables are "always" independent of the rest of the variables which is not true. The choice equivalent to Table 6.2(B) is the one followed because, as mentioned earlier, we do not add a product node unless elements of the disjoint variable groups are uncorrelated. Therefore, we start with a sum node in a similar fashion to Table 6.2(B) (lines 15 & 16), then follow it with a product node (line 18). Note that this results in a sum node with exactly two branches but they can be split through further sum nodes again later if there are further clusters involved which means that the number of clusters can practically be any number depending on how many clusters the algorithm recursively finds.

The algorithm proceeds recursively on each of the three submatrices until one of the two previously mentioned base cases is reached or the base case $stop = true$ is reached. The result in the latter is a submatrix that is approximately of rank-one and we have a *multivariate*

distribution leaf node. We have a submatrix but we still refer to it as a multivariate leaf node because the solution in this case is straightforward. We know this is a rank-one submatrix; i.e. variables are not independent because had there been any independence, the matrix would have been of a higher rank. Therefore, we can proceed with a sum node containing as many branches as number of instances in the submatrix, then a product node for each instance on its own. Even if this seems to be big of a change with a large submatrix, but because we know that a submatrix in this case is of rank-one, we are confident enough that there is no variable independence and that searching for such independences would cause huge loss of run-time without a gain as per the objective function. A summary of the algorithm is provided in Algorithm 2. The $R1Biclus$ subroutine extracts the values $M1$, $N1$, $u1$, $v1$ and $\sigma 1$ first as the outcome of its main loop (lines 29-37). Unlike the original R1D algorithm of Biggs et al. [2008], we do not downdate the matrix here. We take the nonzero indices of $u1$ and $v1$, which are $M1$ and $N1$, as a bicluster on their own. This means that we apply the biclustering based on the resulting values of $u1$ and $v1$ deciding to put those that are nonzero in one bicluster. One of the main advantages of R1D compared to other SVD-based methods is that the rank-one submatrix (bicluster in our case) is searched for and submatrices are compared to one another in an optimisation subproblem of its own [Biggs et al., 2008]. This makes it more convenient for use in splitting variables/instances. Variable splitting should be based on correlation, not similarity. The returned bicluster adapts in case one variable is changing exactly in the opposite direction of another as in this case both have nonzero values. Moreover, both have the same absolute nonzero value of $v1$. For example, look at the following data matrix:

3 -3

3 -3

7 -7

7 -7

The resulting $v1$ vector in this case is $[0.7071\ -0.7071]$ and its absolute is $[0.7071\ \ 0.7071]$. We pick the nonzero positions, and so both variables are picked in one bicluster. This is inline with variable splitting in SPN and in graphical models in general as these values are obviously correlated.

Algorithm 2 SPN Learning (SPN-R1DBiclus)

Input: $A \in R^{m1 \times n1}, \gamma > 1$
Output: SPN representing A

SPN = SPNR1dBiclus(A)

Function SPNR1dBiclus(B)
Let r = $\{1, 2, ..., \#\ rows\ of\ B\}$
Let c = $\{1, 2, ..., \#\ columns\ of\ B\}$
if $|c| = 1$ **then**
  return univariate distribution as a leaf
end if
if $|r| = 1$ **then**
  return handle c leaf univariate distributions
end if
[MB, MC, MD, $Stop$] = R1Biclus(B)
if $Stop$ = true **then**
  return handle multivariate distribution(B)
else
  Let ME = Augmented Matrix(MB, MC)
  Make sum node from MD & ME
  Call SPNR1dBiclus(MD)
  Make product node from MB & MC
  Call SPNR1dBiclus(MC)
  Call SPNR1dBiclus(MB)
end if
End Function

Function [MB, MC, MD, $Stop$] = R1Biclus(B)
Select $j_0 \in \{1, 2, ..., n1\}$ to maximise $\|A(:, j_0)\|$
$M1 = \{1, 2, ..., m1\}$
$N1 = \{j_0\}$
$\sigma 1 = \|A(:, j_0)\|$
$u1 = \frac{A(:, j_0)}{\sigma 1}$
repeat
  $v1 = A(M1, :)^T u1(M1)$
  $N1 = \{j : \gamma\ v1(j)^2 - \|A(M1, j)\|^2 > 0\}$
  $v1(N1) = \frac{v1(N1)}{\|v1(N1)\|}$
  $u1 = A(:, N1)v(N1)$
  $M = \{i : \gamma\ u1(i)^2 - \|A(i, N1)\|^2 > 0\}$
  $\sigma 1 = \|u1(M1)\|$
  $u1(M1) = \frac{u1(M1)}{\sigma 1}$
until $M1$, $N1$, $u1$, $v1$, $\sigma 1$ do not change
$MB = B(M1, N1)$
if $MB = B$ **then**
  $Stop = true$
else
  $Stop = false$
end if
$allColsExN1 = \{1, 2, ..., n1\} \setminus N1$
$allCols = \{1, 2, ..., n1\}$
$allRowsExM1 = \{1, 2, ..., m1\} \setminus M1$
$MC = B(M1, allColsExN1)$
$MD = B(allRowsExM1, allCols)$
End Function

In addition to being used as an SPN structure learning algorithm, $R1Biclus$ is a novel clustering and biclustering algorithm. It can be used for instance clustering by splitting elements instances based on being equivalent to zero or nonzero positions in the $u1$ vector, then doing the same for each cluster on its own. $R1Biclus$ can be used as a biclusteirng algorithm in a manner very similar to what we do here, but based on similarity in both the first and second singular vectors. Biclustering has many applications in biological data analysis. $R1Biclus$ has an advantage over other SVD clustering algorithms, which is it is less prone to perturbations by noise because it computes the dominant singular vectors of a submatrix rather than the full matrix. An article classification application of the $R1D$ algorithm in Biggs et al. [2008], by which $R1Biclus$ was motivated, shows that the original algorithm is a good material for learning applications.

6.6 Experiments

In order to model $P(F|I)$ in the BIF model described in Chapter 5, density estimators were used. Here, the impact of computing $P(F|I)$ using SPN whose structure is learnt by SPN-R1DBiclus and using an SPN whose structure is learnt by SPN-Gens is compared to the density estimation techniques used in Chapter 5 (multivariate Gaussian, Kernel density estimation and Gaussian Copula with KDE Marginals). The evaluation measures used are the ultimate bag accuracy and log-likelihood value, same as in Chapter 5. In Adel and Ghodsi [2014], a more comprehensive set of experiments was performed. This set of experiments contained log-likelihood and conditional log-likelihood values comparisons, learning run-time comparisons on image datasets with larger size and dimension, an image completion task and a handwritten digit recognition application. The results of these experiments demonstrated that SPN-R1DBiclus leads to improvements in terms of log-likelihood and conditional log-likelihood values, learning accuracy as well as structure learning run-time.

As noted in Chapter 5, the MUSK1 dataset [Dietterich et al., 1997] has 168 discrete features. It contains 92 bags and 476 instances. Table 6.3 shows the results of the BIF model on the MUSK1 dataset. Bag accuracy as well as log-likelihood values obtained by the SPN

TABLE 6.3: Results of the MUSK1 dataset after using SPNs.

MUSK1	BIF: $B \rightarrow I \rightarrow F$ $_m$						
Rnd: 0.33	⫫Gauss	⫫Cop.	Gauss	Cop.	KDE	SPN-R1DBiclus	SPN-Gens
Bag Acc.	**86.5**%	**84.4**%	69.6%	.641%	**77.2**%	**91**%	**90.1**%
Log Lik.	-14921	-18437	-45815	-51031	-33591	*-11894*	-12200

structure learning algorithm SPN-R1DBiclus are better than those obtained using density estimators and using an SPN learnt by SPN-Gens instead.

As per learning times of both SPN structure learning algorithms, learning time of SPN-R1DBiclus is 0.4 the average learning time of SPN-Gens. The introduction of the multivariate leaf node concept is one key factor behind the difference in learning times in favour of SPN-R1DBiclus.

6.7 Summary

With the specific goal here being to represent the joint distribution $P(F|I)$ of the BIF-structure efficiently, we presented a new SPN structure learning algorithm that provides a compact and tractable representation of the joint feature probability distribution (given instance feature) of the MUSK data. The constructed SPN does not depend on local greedy data splittings as it globally splits the data based on a biclustering algorithm. The algorithm leads to improvements in the MUSK1 bag accuracy and log-likelihood, over density estimators ((multivariate Gaussian, Kernel density estimation and Gaussian Copula with KDE Marginals)) and SPN structure learning represented by the state-of-the-art SPN structure learning algorithm developed by Gens and Domingos [2013]. The algorithm is also far superior in terms of SPN structure learning speed.

In addition to being an SPN structure learning algorithm, the core part of the presented algorithm represents a versatile SVD-based biclustering algorithm that can be used for both clustering and biclustering. Here we showed results that are relevant to the MIL data. A more comprehensive set of results and algorithms to compare with is shown in Adel and Ghodsi [2014], where state-of-the-art results of learning, inference on image datasets, image

completion and a handwritten digit classification application are provided by the presented algorithm.

Chapter 7

Conclusions

Research in this thesis has focused on learning problems where partially labelled data are available. More specifically, data that come in the form of bags of instances where labels are available for bags, but not for instances. This is the form of the electromyographic (EMG) data which was collected for muscle diagnosis and which represents the main application. The main evaluation measure used is the bag accuracy. The premise of the introduced learning paradigms is to exploit prior information and assumptions about the partially labelled training data so that further labelling information can be obtained before building the predictor or classifier accordingly. By gaining further labelling information, the mapping from instances to labels provided by the classifier improves.

Two probabilistic paradigms are introduced as approaches for bags-of-instances data. The first approach is a discriminative weakly supervised learning approach where unlabelled data are weakly annotated via first performing spectral graph-theoretic grouping and then assigning weak annotations based on the labelled data within the associated spectral groups. A number of similarity graph models for spectral grouping, including two introduced similarity graph models, are explored to investigate their performance in handling different types of data. The second approach introduced as a solution for bags-of-instances data is a generative modelling approach for MIL that casts the problem as an MIL problem and allows for expert domain knowledge to be incorporated intuitively which leads to good inductive bias and consequently high bag accuracy.

A state-of-the-art solution to the muscle classification problem has been provided by the MIL generative modelling and weakly supervised learning.

One advantage of generative models is that they can be used as a simulation tool as they model the joint distribution of data and label. As such, MIL generative models can be used in EMG data simulation, which represents one path for future work in EMG and muscle analysis.

When number of bags of each label is small, the spectral graph-theoretic grouping phase is not affected as long as number of instances is not small, because spectral grouping collects all unlabelled instances of the same bag label in one graph (as if it is one bag) before starting to group them on the graph. However, generative MIL models need some population of bags so that they can generalise and assign probabilities related to the bag label B on the base of this population.

MIL generative models have the advantage of being intrinsically capable of exploiting unlabelled instances as they can be treated as latent variables, which was the case with the MIL instance labels. On the other hand, the weakly supervised learning approach does not have this advantage. Spectral graph-theoretic grouping makes it possible to exploit unlabelled data in learning, but this is due to the assumptions of the EMG muscle data that make clustering a sound option. For example, the MUSK dataset, where many positive bags have a very small number of positive instances that can not be considered a group or a cluster, does not represent good material for a grouping-based procedure like spectral graph-theoretic grouping. However, more generically, the weakly supervised learning approach has spectral graph-theoretic grouping as one phase and replacing this phase with another phase that is more sound according to the data and problem at hand can be useful as long as the added phase reliably assigns weak annotations to unlabelled data.

As was the case with MIL generative models, the need to model complex probability distributions efficiently is a must for probabilistic models especially with datasets of large size and/or dimension. We introduce an SPN structure learning algorithm that results in a compact and tractable representation of the joint feature probability distribution of the MUSK data. This leads to overall higher bag accuracy of the resulting MIL generative model. The

algorithm aims at learning an SPN structure by concurrently splitting the data matrix in two directions using a biclustering algorithm which aims at making the algorithm model coherent chunks of the data matrix as subSPNs and consequently representing distributions (feature distribution given instance label in the MIL generative model case) with high-fidelity models.

Bibliography

T. Adel and A. Ghodsi. Structure learning of sum-product networks via rank-one downdate biclustering. *Under review*, 2014.

T. Adel, B. Smith, and D. Stashuk. Muscle categorization using pdf estimation and naive bayes classification. *IEEE Engineering in Medicine & Biology Society (EMBC)*, pages 2619–2622, 2012.

T. Adel, R. Urner, B. Smith, D. Stashuk, and D. Lizotte. Generative multiple-instance learning models for quantitative electromyography. *Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 332–339, 2013.

T. Adel, A. Wong, and D. Stashuk. A weakly supervised learning approach based on spectral graph-theoretic grouping. *To be submitted in April 2014*, 2014.

S. Aksoy and R. M. Haralick. Graph-theoretic clustering for image grouping and retrieval. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1999.

K. Ali, D. Hasler, and F. Fleuret. Flowboost - appearance learning from sparsly labeled video. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

E. Alpaydin. *Introduction to Machine Learning*. The MIT Press, 2nd edition, 2010.

S. Andrews and T. Hofmann. Multiple-instance learning via disjunctive programming boosting. *Advances in Neural Information Processing Systems (NIPS)*, pages 65–72, 2003.

S. Andrews, T. Hofmann, and I. Tsochantaridis. Multiple instance learning with generalized support vector machines. *Conference of the American Association for Artificial Intelligence (AAAI)*, pages 943–944, 2002a.

S. Andrews, I. Tsochantaridis, and T. Hofmann. Support vector machines for multiple-instance learning. *Advances in Neural Information Processing Systems (NIPS)*, pages 561–568, 2002b.

H. Arora, N. Loeff, D. Forsyth, and N. Ahuja. Unsupervised segmentation of objects using efficient learning. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

A. Asuncion and D. Newman. Uci machine learning repository. 2007. URL http://www.ics.uci.edu/~mlearn/MLRepository.html.

P. Barkhaus and S. Nandedkar. Recording characteristics of the surface EMG electrodes. *Muscle & Nerve*, pages 1317–1323, 1994.

S. Ben-David, U. Von Luxburg, and D. Pal. A sober look at clustering stability. *Conference on Learning Theory (COLT)*, pages 5–19, 2006.

S. Ben-David, N. Srebro, and R. Urner. Universal learning vs. no free lunch results. *Philosophy and Machine Learning Workshop NIPS*, 2011.

Y. Bengio. Learning deep architectures for ai. *Foundations and Trends in Machine Learning*, 2009.

A. Bergamo and L. Torresani. Exploiting weakly-labeled web images to improve object classification: A domain adaptation approach. *Advances in Neural Information Processing Systems (NIPS)*, pages 181–189, 2010.

M. Biggs, A. Ghodsi, and S. Vavasis. Nonnegative matrix factorization via rank-one downdate. *International Conference on Machine Learning (ICML)*, 25, 2008.

C. M. Bishop and J. Lasserre. Generative or discriminative? getting the best of both worlds. *Bayesian Statistics*, 8:3–23, 2007.

C. Blake, E. Keogh, and C. Merz. Uci repository of machine learning databases. 1998.

M. Blaschko, A. Vedaldi, and A. Zisserman. Simultaneous object detection and ranking with weak supervision. *Advances in Neural Information Processing Systems (NIPS)*, 2010.

C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller. Context-specific independence in bayesian networks. *Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 115–123, 1996.

N. Campbell and J. Reece. *Biology.* The MIT Press, 6th edition, 2001.

Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 27:1–27, 2011.

O. Chum and A. Zisserman. An exemplar model for learning object classes. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

F. Chung. *Spectral graph theory.* AMS Press, 1997.

C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

D. J. Crandall and D. Huttenlocher. Weakly supervised learning of part-based spatial models for visual object recognition. *European Conference on Computer Vision (ECCV)*, 2006.

D. L. Davies and D. W. Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2:224–227, 1979.

N. de Freitas and H. Kück. Learning about individuals from group statistics. *Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 332–339, 2005.

A. Dennis and D. Ventura. Learning the architecture of sum-product networks using clustering on variables. *Advances in Neural Information Processing Systems (NIPS)*, 25, 2012.

T. G. Dietterich, R. H. Lathrop, and T. Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1-2):31–71, 1997.

P. Domingos. The role of occam's razor in knowledge discovery. *Data mining and knowledge discovery*, 3(4):409–425, 1999.

D. Dumitru, A. Amato, and M. Zwarts. *Electrodiagnostic Medicine*. 2nd edition, 2002.

Murat Dundar, Glenn Fung, Balaji Krishnapuram, and R. Bharat Rao. Multiple-instance learning algorithms for computer-aided detection. *IEEE Transactions on Biomedical Engineering*, 55(3):1015–1021, 2008.

C. Farkas, A. Hamilton-Wright, H. Parsaei, and D. Stashuk. A review of clinical quantitative electromyography. *Critical Reviews in Biomedical Engineering*, 38(5):467–485, 2010.

B. Flury and H. Riedwyl. *Angewandte multivariate Statistik*. MIT Press, 1983.

J. R. Foulds and E. Frank. A review of multi-instance learning assumptions. *Knowledge Engineering Review*, 25(1):1–25, 2010.

J. R. Foulds and P. Smyth. Multi-instance mixture models. *SIAM International Conference on Data Mining (SDM)*, pages 606–617, 2011.

C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral grouping using the Nyström method. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 26 (2):1–12, 2004.

T. Gärtner, P. A. Flach, A. Kowalczyk, and A. J. Smola. Multi-instance kernels. *International Conference on Machine Learning (ICML)*, pages 179–186, 2002.

S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias-variance dilemma. *Neural Computation*, 4:1–58, 1992.

R. Gens and P. Domingos. Discriminative learning of sum-product networks. *Advances in Neural Information Processing Systems (NIPS)*, 25, 2012.

R. Gens and P. Domingos. Learning the structure of sum-product networks. *International Conference on Machine Learning (ICML)*, 30, 2013.

G. H. Golub and C. F. Van Loan. *Matrix computations*. Johns Hopkins University Press, 1996.

M. Guillaumin, J. Verbeek, and C. Schmid. Multimodal semi-supervised learning for image classification. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

Y. Guo, A. Korhonen, I. Silins, and U. Stenius. Weakly supervised learning of information structure of scientific abstracts - is it accurate enough to benefit real-world tasks in biomedicine? *Bioinformatics*, 27(22):3179–3185, 2011.

L. Hagen and A. Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE Transactions on Computer-Aided Design*, 11(9):1074–1085, 1992.

T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning*, volume 2. 2009.

M. Hein and U. von Luxburg. Similarity graphs in machine learning. *Machine Learning Summer School, MPI Biological Cybernetics*, 2007.

G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.

T. Hofmann. Probabilistic latent semantic analysis. *Conference on Uncertainty in Artificial Intelligence (UAI)*, 15:289–296, 1999.

T.-K. Huang, R. C. Weng, and C.-J. Lin. Generalized Bradley-Terry models and multi-class probability estimates. *Journal of Machine Learning Research (JMLR)*, 7:85–115, 2006.

G. James, D. Witten, T. Hastie, and R. Tibshirani. *An Introduction to Statistical Learning*, volume 1. 2013.

A. Joulin and F. Bach. A convex relaxation for weakly supervised classifiers. *International Conference on Machine Learning (ICML)*, 2012.

F. Keinosuke. *Introduction to statistical pattern recognition*, volume 1. 1990.

D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *International Conference on Machine Learning (ICML)*, 2001.

S.-W. Lee, H. Min-Oh, and Z. Byoung-Tak. Online incremental structure learning of sumproduct networks. *In Neural Information Processing*, 2013.

Y. J. Lee and K. Grauman. Learning the easy things first: Self-paced visual category discovery. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

C. Leistner, M. Godec, S. Schulter, A. Saffari, and H. Bischof. Improving classifiers with weakly-related videos. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

Y. Li, I. Tsang, J. Kwok, and Z. Zhou. Convex and scalable weakly labeled svms. *Journal of Machine Learning Research (JMLR)*, 2013.

J. Malmivuo and R. Plonsey. *Bioelectromagnetism principles and applications of bioelectric and biomagnetic fields*. Oxford University Press, 1995.

M. I. Mandel and D. P. Ellis. Multiple-instance learning for music information retrieval. *International Society for Music Information Retrieval (ISMIR)*, pages 577–582, 2008.

O. Maron and A. L Ratan. Multiple-instance learning for natural scene classification. *International Conference on Machine Learning (ICML)*, pages 341–349, 1998.

A. Ng and M. I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Advances in Neural Information Processing Systems (NIPS)*, 2001.

A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: analysis and an algorithm. *Advances in Neural Information Processing Systems (NIPS)*, 14:849–856, 2002.

H. Parsaei, D. Stashuk, and T. Adel. Decomposition of intramuscular emg signals using a knowledgebased certainty classifier algorithm. *IEEE Engineering in Medicine & Biology Society (EMBC)*, pages 6208–6211, 2012.

R. Peharz, B. C. Geiger, and F. Pernkopf. Greedy partwise learning of sum-product networks. *In Machine Learning and Knowledge Discovery in Databases*, 8189:612–627, 2013.

G. Pfeiffer. The diagnostic power of motor unit potential analysis: An objective bayesian approach. *Muscle & Nerve*, 22:584–591, 1999.

G. Pfeiffer and K. Kunze. Discriminant classification of motor unit potentials (MUPs) successfully separates neurogenic and myopathic conditions. *Electromyography and Motor Control*, 97:191–207, 1995.

L. Pino. *Neuromuscular Clinical Decision Support using Motor Unit Potentials Characterized by Pattern Discovery.* PhD Thesis, Systems Design Engineering, Univ. of Waterloo, 2008.

H. Poon and P. Domingos. Sum-product networks: A new deep architecture. *Conference on Uncertainty in Artificial Intelligence (UAI)*, 27, 2011.

A. Prest, C. Leistner, J. Civera, C. Schmid, and V. Ferrari. Learning object class detectors from weakly annotated video. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

R. Rahmani and S. A. Goldman. Missl: multiple-instance semi-supervised learning. *International Conference on Machine Learning (ICML)*, pages 705–712, 2006.

C. J. Van Rijsbergen. *Information retrieval.* Butterworth, 2nd edition, 1979.

A. Robergs and S. Roberts. *Exercise physiology.* Mosby, 1996.

A. Rooshenas and D. Lowd. Learning sum-product networks with direct and indirect variable interactions. *International Conference on Machine Learning (ICML)*, 31, 2014.

D. Roth. On the hardness of approximate reasoning. *Artificial Intelligence*, 82:273–302, 1996.

S. Sabato, N. Srebro, and N. Tishby. Reducing label complexity by learning from bags. *Journal of Machine Learning Research (JMLR)*, 9:685–692, 2010.

B. Settles, M. Craven, and S. Ray. Multiple-instance active learning. *Advances in Neural Information Processing Systems (NIPS)*, 2007.

J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 22(8):888–905, 2000.

B. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall/CRC, 1998.

A. Sklar. Fonstions de répartition à n dimensions et leurs marges. *Publications de l'Institut de Statistique de l'Université de Paris*, 8:229–231, 1959.

D. Stashuk. Mean, median and mode estimation of motor unit action potential templates. *IEEE Engineering in Medicine & Biology Society (EMBC)*, 4:1498–1499, 1996.

D. Stashuk. Decomposition and quantitative analysis of clinical electromyographic signals. *Medical Engineering and Physics*, 21(6):389–404, 1999.

M. Stikic and B. Schiele. Activity recognition from sparsely labeled data using multi-instance learning. *Location and Context Awareness (LoCA)*, pages 156–173, 2009.

M. Stoer and F. Wagner. A simple min-cut algorithm. *Journal of the ACM (JACM)*, 44(4): 585–591, 1997.

Q. Tao. Making efficient learning algorithms with exponentially many features. *University of Nebraska, PhD Thesis*, 2004.

Q. Tao, S. D. Scott, N. V. Vinodchandran, and T. T. Osugi. Svm-based generalized multiple-instance learning via approximate box counting. *International Conference on Machine Learning (ICML)*, 2004.

P. A. Viola, J. C. Platt, and C. Zhang. Multiple instance boosting for object detection. *Advances in Neural Information Processing Systems (NIPS)*, 2005.

U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

J. Wang and J. Zucker. Solving the multiple-instance problem: A lazy learning approach. *International Conference on Machine Learning (ICML)*, pages 1119–1126, 2000.

J. Winn and N. Jojic. Locus: learning object classes with unsupervised segmentation. *International Conference on Computer Vision (ICCV)*, 2005.

T. Wu, C. Lin, and R. Weng. Probability estimates for multi-class classification by pairwise coupling. *Journal of Machine Learning Research (JMLR)*, 2004.

Z. Wu and R. Leahy. An optimal graph theoretic approach to data clustering: theory and its application to image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 15(11):1101–1113, 1993.

X. Xu and E. Frank. Logistic regression and boosting for labeled bags of instances. *Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, pages 272–281, 2004.

S. H. Yang, H. Zha, and B. G. Hu. Dirichlet-bernoulli alignment: A generative model for multi-class multi-label multi-instance corpora. *Advances in Neural Information Processing Systems (NIPS)*, pages 2143–2150, 2009.

D. Yates, D. S. Moore, and D. S. Starnes. *The practice of statistics*. Freeman, 2nd edition, 2003.

D. Zennaro, P. Welling, V. Koch, G. Moschytz, and T. Laubli. A software package for the decomposition of long-term multichannel emg signal using wavelet coefficients. *IEEE Transactions on Biomedical Engineering*, pages 58–69, 2003.

Qi Zhang, S. A. Goldman, W. Yu, and J. E. Fritts. Content-based image retrieval using multiple-instance learning. *International Conference on Machine Learning (ICML)*, pages 682–689, 2002.

Z. Zhou and J. Xu. On the relation between multi-instance learning and semi-supervised learning. *International Conference on Machine Learning (ICML)*, pages 1167–1174, 2007.