ESCUELA TÉCNICA SUPERIOR DE INGENIEROS
INFORMÁTICOS

UNIVERSIDAD POLITÉCNICA DE MADRID

TESIS DOCTORAL
**IMAGE SEGMENTATION WITH BOOSTED
MULTI-CLASS CLASSIFIERS**

presentada en la
ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INFORMÁTICOS
de la
UNIVERSIDAD POLITÉCNICA DE MADRID
para la obtención del
GRADO DE DOCTOR EN INFORMÁTICA

AUTOR:
Kendrick Cetina Rojas

DIRECTORES:
José M. Buenaposada Biencinto y Luis Baumela Molina

MADRID, 2018

Image Segmentation with Boosted Multi-class classifiers

Kendrick Cetina Rojas

Agradecimientos

Me gustaría primeramente agradecer a Luis, por haber despertado en mí el interés por la visión por computadora y el aprendizaje automático. Muchas gracias por haber sido el director de esta tesis y gracias por las horas de esfuerzo y dedicación a este trabajo. Muchas gracias por haber abogado por este trabajo ante editores y autores. Y por haber ofrecido muchísima ayuda en mi formación. Muchas gracias.

También quiero agradecer a JM por sus comentarios e ideas que sin duda han llevado este trabajo a un mejor nivel del que yo había pensado. Gracias por el interés en este trabajo. Gracias por el ánimo dado en esos momentos en los que los resultados parecían no salir a flote y gracias por las conversaciones y correos interesantes. Me han hecho crecer no solamente en lo académico. Gracias también por las veces que me acercaste a la renfe desde la facultad. Gracias.

A Pablo, my master, gracias. Por haber sido de mucha ayuda en mis primeros años de doctorado, gracias por las veces que me explicaste la teoría tras la práctica. Muchas gracias por haber sido parte desde el inicio de esta tesis. Gracias por la información que facilitó mi trabajo y por responder a los correos incluso de temas tontos, gracias. Cuando alguien me pregunta si quiero escuchar algo interesante, ahora mi respuesta es: Siempre.

También quiero agradecer a Antonio por haber sido un fundamento a través de sus algoritmos y por el ánimo dado en el laboratorio.

Gracias, Roberto, por el apoyo moral y por las conversaciones interesantes a la hora de comer.

Gracias, Xesca la aportación de los descriptores elípticos. Gracias a Pascal Fua por los stacks de imágenes.

A mis padres, muchas gracias. Todo lo bueno que tengo viene de ustedes. Todo lo malo son cosas que habré recogido por el camino.

Y a Pamela, por haber salido siempre al rescate y haberme aguantado. Muchas gracias.

Resumen

Segmentar una imagen consiste en dividirla en un conjunto de regiones o segmentos, generalmente dotados de alguna semántica. Es, por lo tanto, uno de los problemas más complejos a los que se enfrenta la visión por computador, pues en el caso más general implica comprender "qué hay en la imagen".

En esta tesis abordamos el problema de la segmentación de imágenes planteando una solución supervisada que se apoya en recientes algoritmos de clasificación multi-clase basados en Boosting. Aunque el planteamiento de la solución es general, nos hemos centrado en el problema de segmentación de imágenes de microscopía electrónica (EM) del cerebro.

Frente a la mayoría de trabajos publicados que segmentan una única estructura, nuestra propuesta parte de la idea de que el análisis simultáneo de varias estructuras cerebrales mejora el rendimiento de los algoritmos de segmentación, pues permite explotar mejor las dependencias entre las clases de estructuras y aprender descripciones de las imágenes que proporcionen mejores resultados de segmentación, tal como se demuestra en la tesis.

En esta tesis proponemos una técnica de segmentación de imágenes que es el estado del arte en algoritmos basados en Boosting para segmentación de dos estructuras cerebrales, sinapsis y mitocondrias, en imágenes EM del cerebro. Para lograr este objetivo realizamos varias aportaciones en diferentes áreas.

Primero, estudiamos los descriptores de características de imagen más populares y compararnos su desempeño en imágenes EM, seleccionando las que producen las mejores métricas de segmentación. Para llegar a este resultado, hemos desarrollado un nuevo algoritmo de selección de características que es capaz de seleccionar el mejor conjunto de características y escalas para describir una pila de imágenes EM determinada.

En segundo lugar, presentamos la *curva Jaccard* como una herramienta para comparar el rendimiento de dos algoritmos de segmentación independientemente de su punto de operación. Usamos esta curva para com-

parar el rendimiento de nuestro enfoque con los mejores algoritmos en la literatura.

Como contribución final de la tesis, y para mejorar aún más la segmentación, desarrollamos un procedimiento general para optimizar los índices de calidad típicos en segmentación de imágenes. Para este fin, usamos resultados recientes de clasificación con costes para encontrar los costes que optimizan el índice deseado. Los experimentos muestran que nuestro procedimiento, utilizando un conjunto de características sencillas, puede mejorar el índice de Jaccard en las segmentaciones de las sinapsis y las mitocondrias, así como las de otras imágenes naturales.

Abstract

The goal of image segmentation is to partition an image into different regions, each of them usually endowed with a semantic meaning (i.e. road, house, car, etc). Hence, it is one of the most complex problems faced by computer vision, since in its most general case it entails understanding "what is in the image."

In this thesis we address the image segmentation problem from a supervised perspective. Our solution is supported by recent multi-class Boosting algorithms. Although we approach the problem from a general view point, our algorithms were conceived to address the problem of segmenting electron micrographs (EM) of brain tissue.

Most state-of-the-art-approaches for this problem segment one single brain structure at a time. Instead, the starting point of our proposal is that the simultaneous analysis of several brain structures improves the performance of the segmentation. This lets the algorithm exploit the dependencies among the different classes of structures and find image descriptions that provide better segmentation results, as we prove in our experiments.

In this thesis, we develop an image segmentation algorithm that is the state-of-the-art Boosting scheme for segmenting two brain structures, mitochondria and synapses, in EM images of the brain. To reach this goal we make several contributions in different areas.

First, we study the most popular image feature descriptors and compare their performance on EM images, selecting the ones that produce the best segmentation metrics. To reach this result we develop a new feature selection algorithm that is able to select the best set of features and scales to describe a given EM image stack.

Secondly, we introduce the *Jaccard Curve* as a tool to compare the performance of two segmentation algorithms independently of their operation point. We use this curve to compare the performance of our approach with the best algorithms in the literature.

As a final contribution of the thesis, and to improve even further the

segmentation, we develop a general procedure to optimize typical image segmentation quality indices. To this end we use recent cost-sensitive classification results to learn the costs that optimize the desired index. The experiments show that our procedure, using a pool of simple features, can improve significantly the Jaccard index in the segmentations of synapses and mitochondria as well as that of other natural images.

Contents

1	Introduction	13
1.1	Contributions of the thesis	16
1.2	Organization of the thesis	18
2	Background	19
2.1	Supervised learning	20
2.1.1	Boosting classifiers	21
2.1.2	Unbalanced vs asymmetric classification problems	27
2.1.3	Segmentation metrics	29
2.2	EM image segmentation	31
3	Channel and scale selection for EM images of the brain	35
3.1	Features	36
3.1.1	Simple Window and Histogram	36
3.1.2	Local Binary Patterns	37
3.1.3	GRIMS	37
3.1.4	Ray Descriptors	39
3.1.5	Difference of Gaussian's	40
3.1.6	Laplacian Of Smoothed Image	40
3.1.7	Eigenvalues of the structure tensor	40
3.1.8	Histogram of Oriented Gradients	41
3.1.9	Radon-Like features	41
3.2	Scales Selection	41
3.3	Experiments	43
3.4	Conclusions	51
4	Multi-class Boosting for segmentation of multiple structures	53
4.1	Image features used	55
4.1.1	Scales selection	55
4.1.2	Feature aggregation	57
4.2	Multi-class boosting with integral channel features	59

CONTENTS

4.2.1	Weighted sampling for Boosting	59
4.2.2	Label regularization	61
4.3	Results and Discussion	62
4.3.1	Quality measure	63
4.3.2	Experiments	63
4.3.3	Results of basic classification algorithm.	65
4.3.4	Classification and regularization results.	68
4.3.5	Ablation analysis.	68
4.3.6	Computational cost.	72
4.4	Conclusions	75
5	Optimizing segmentation using costs	77
5.1	Optimizing metrics with costs	79
5.2	Experiments and Discussion	81
5.2.1	MSRCv2 dataset	81
5.2.2	Mitochondria and synapse segmentation	84
5.3	Conclusions	86
6	Conclusions	89
6.1	Publications derived from the thesis	90

List of Figures

1.1	Example of segmentation of an image (a) from the MSRCv2 5.2.1 dataset with the labels (b) associated to the classes like <i>plane</i> (red) and <i>sky</i> (grey).	13
1.2	Neural tissue image stack acquired from the Hippocampus of a rat using EM Microscopy. (a) depicts 165 slices of the stack. (b) shows the map of mitochondria in (a).	15
2.1	EM image (a) with labels for structures: synapse (b) and mitochondria (c). We can see the imbalance of data belonging to class synapse against the background. In (d) we can see the noisy prediction of a classifier trying to segment mitochondria.	28
2.2	Jaccard similarity coefficient. (left) in blue the ground truth segmentation of a mitochondrion, in red the result obtained with an automated segmentation algorithm; TP is the intersection of red and blue regions, i.e. the correctly segmented piece of mitochondrion; FN is the only blue area, i.e. the part of mitochondrion segmented as background; FP is the only red area, i.e. the background segmented as mitochondrion; TN is the rest of the image, i.e. the correctly segmented background; (right) representation of the Jaccard coefficient as the relation between the purple and green areas.	30
2.3	Example of a ROC curve over artificial data from [82]. Higher values of AUC represents better segmentation.	31
3.1	A patch from the stack (a) and its labels (b), class mitochondria in grey, class synapses in white.	36
3.2	Simple window and histogram descriptors.	37

LIST OF FIGURES

3.3	Depending on the values of the neighbouring pixels, the LBP generates a binary code that can be converted to a real value. In this case the value would be: $0 \times 1 + 0 \times 2 + 1 \times 4 + 1 \times 8 + 0 \times 16 + 0 \times 32 + 0 \times 64 + 1 \times 128 = 148$	37
3.4	Function c returns the position $\mathbf{c}$ from the nearest edge or contour of the image $\mathbf{I}$ to the position $\mathbf{m}$ in direction defined by the angle θ	39
3.5	PIBoost <i>separator</i> in a three class problem. It is composed of three weak-learners (S_1, S_2, S_3) separating each class, e.g. C_1 :mitochondrion, C_2 :synapse, C_3 :background, from the rest.	42
3.6	Comparative segmentation results of Random Forest classifier (RF) and Gaussian Classifier (GC) with tested features: GRIMS, simple window, histogram, difference of Gaussians (DoG), Laplacian of Gaussians (LoG) and HOG.	47
3.7	Jaccard index of synapse and mitochondria segmentation. The top figure shows results with a random forest classifier, the bottom figure shows results with a Gaussian classifier.	48
3.8	ROC Curves. ROC Curves of class Mitochondria. Dotted lines are the experiments done with a Gaussian classifier (GC) and continuous lines with a random forest classifier (RF).	49
3.9	ROC Curves of class Synapse. Dotted lines are the experiments done with a Gaussian classifier (GC) and continuous lines with a random forest classifier (RF).	50
4.1	Hippocampus rat neural tissue SBFEM image. (a) image stack; (b) 3D reconstruction of mitochondria in the stack; (c) ground truth labels of mitochondria (blue) and synapses (red) in the first slice. Membrane labels (green) have been included only for illustrative purposes.	54
4.2	Raw image and four image description features used in our methodology. In the raw image we highlight with blue and red color a synapse and three vesicles respectively. We can appreciate the elongated shape of the synapse and the small circular shape of the vesicles, next to the synapse.	56
4.3	Elliptical descriptor. (left) image showing the kernel as grey values, with both radii(r_1, r_2) and the thickness (w) parameters over-imposed; (enter) image of a vesicle-like structure; (right) response obtained when convolving the vesicle image with the elliptical descriptor kernel.	56

LIST OF FIGURES

4.4	Feature extraction process. We apply a set of filters at scales $\sigma_1, \dots, \sigma_n$ to each stack slice. (a) two filters for one slice, s_{000} left, λ_3 right; (b) a feature of voxel V_i is the sum of the values of one box (in blue). The feature vector of V_i is the concatenation of several hundreds of such features.	58
4.5	Example of a Jaccard curve obtained with the results of the segmentation of two synapses. We selected a hard-to-segment patch to see the segmentation improvement with different thresholds. In the vertical axis we represent the Jaccard (JAC), whereas in the horizontal the percentage of pixels segmented as background. We can see how the higher the threshold (θ) the less elements are segmented as synapse. . .	64
4.6	Membrane detail.	66
4.7	Mitochondria, synapses and membranes JCCs for the Hippocampus (a), Somatosensory (b) and Cerebellum (c) SB-FEM image stacks. In each curve we show with a red dot the zero threshold operating point for Boosting classifiers and MAP point for the Bayesian one. JCCs let us compare the segmentation performance regardless of the operating point. The higher the curve the better the segmentation algorithm.	73
4.8	Segmentation results for each stack and algorithm after regularization.	74
5.1	Qualitative results of MSRCv2 dataset segmentations before and after our optimization.	82
5.2	Cost evolution of the MSRCv2 data set for F1-Score and Jaccard index	84
5.3	Result of segmenting mitochondria (top row) and synapses (bottom row) of a slice of Hippocampus stack before and after Jaccard index optimization.	85
5.4	Cost evolution of the Hippocampus data set for F1-Score and Jaccard index	86

LIST OF FIGURES

List of Tables

3.1	Learning and prediction times	45
4.1	Vesicle descriptor parameters and GRIMS scales selected for each data set.	57
4.2	Comparison of the sampling methods used in our experiments. Jaccard index results from RUSBoost and Weighted sampling approaches.	61
4.3	Data sets used in the experiments. We denote the annotated labels in each data set as synapses (Syn), mitochondria (Mit) and membranes (Mem). We select train and test sets from consecutive slices in the stack. We use approximately half of each stack for training and the rest for testing.	65
4.4	Quantitative segmentation results for mitochondria, synapses and membrane on the Hippocampus, Somatosensory Cortex and Cerebellum stacks evaluated with the Jaccard index.	69
4.5	Quantitative segmentation results for membrane Cerebellum stack evaluated with the <i>Rand F-score</i> ($\mathcal{F}_r$) and <i>Information Theoretic F-score</i> ($\mathcal{F}_{it}$) metrics.	70
4.6	Results of table 4.8 excluding voxels. The performance improved by not taking into account voxels and slices in the borders in the edges of the test stack during evaluation.	70

LIST OF TABLES

4.7	The result of excluding voxels and slices in evaluation. In (a) the performance of synapse segmentation is increased over 0.15 points in Jaccard curves obtained by excluding first 10 and last 10 slices and 10 voxels from the top and bottom of the test stack. The highest point of CCBoost Jaccard curve can be obtained by thresholding a posteriori probabilities of AdaBoost output with threshold ≈ 8 , which reach the performance presented in [5]. We can compare the curves in (a) with the equivalent curves without voxels exclusion in figure 4.7(a). A qualitative result of segmentation of 2 synapses in one of 7th slice of the stack is depicted in (b). CCBoost output is noisy and after graph cut (GC) regularization the classification improves. For PIBoost, the segmentation is less noisy. This explains the high improvement in performance of the segmentation after regularization of table 4.4 for hippocampus stack. In both PIBoost and CCBoost after regularization a entire synapse is missing.	71
4.8	Hippocampus segmentation results (Jaccard) for different classifier, features and GRIMS scales.	72
5.1	MSRCv2 dataset results before and after Jaccard index optimization.	83
5.2	Results of BadaCost before and after optimization for Jaccard index.	85

1

Introduction

One of the traditional research areas of computer vision is the partitioning of digital images into segments or subsets of pixels, representing different structures of the scene or classes of objects [96]. To be more specific, segmentation is the process of assigning a label to every pixel in an image. (See figure 1.1). It is used either to find boundaries, curves, objects or regions [4]. There are many applications of segmentation like: object detection [63], extraction of homogeneous regions or superpixels [87], label of drive and no-drive regions for autonomous cars [8], etc.

Figure 1.1: Example of segmentation of an image (a) from the MSRCv2 5.2.1 dataset with the labels (b) associated to the classes like *plane* (red) and *sky* (grey).

There are many approaches to segment an image, the simplest way could be to threshold the intensity values of the pixels of the image, this would result in a binary segmentation. Meaning that a pixel could belong to one of two classes (above or below the threshold). Nonetheless, there are more refined methods to segment an image that allow the possibility of distinguishing more than two classes and assign a pixel one or more la-

bels, among a set of predefined ones. There are different types of segmentation problems. If a pixel can take more than one label then it is called a multi-label segmentation problem. If the pixel can only take one of several labels, then it is called a multi-class segmentation problem. If a pixel can only take one of two values, it is called a binary segmentation. When the structures to segment become heterogeneous (in color, shape, texture, etc.) and the complexity of segmentation increases, the problem becomes less straightforward and we need to take advantage of complex features from the images to learn a model able to identify the classes of interest. The pixels that are assigned the label y should have a set of common features that could be texture, color, shape, etc. The process of obtaining the most informative features of a class is called *feature extraction*, and the process of learning a model is a machine learning technique called *classification*.

In the main application of this work we will be focusing on one of the most challenging problems of science nowadays, which is the analysis of medical images to create a digital reconstruction of the brain. The goal is to comprehend the architecture of the brain by analyzing tissue and constructing detailed maps of neuron structures. This will help to better understand functions and pathologies of the brain [20]. A key step to achieving this goal is the segmentation of image stacks acquired with automated electron microscope techniques [25, 50]. The complexity and the large amount of images in a stack make automatic segmentation the only feasible solution.

Figure 1.2 shows an example of one of the datasets we will be working with. Images piled in a stack allow the construction of 3D representation of brain structures, and two of the most analysed structures are mitochondria and synapses. The neurological interest of these structures relies on the information given by the morphology, distribution and its quantification [10]. Mitochondria provide energy to the cell. They are responsible for many essential cellular functions having great importance in cellular physiology [14]. Meanwhile, synapses are in charge of the communication between neurons, which is crucial for cognition.

The performance of the segmentation is evaluated by different metrics, which should address an important issue, the inherent asymmetry of the segmentation process, because of the highly disproportion of samples in classes. The *background* is often the class with more samples. In problems like the one we address in this thesis the background can be more than the 90% of the data samples. We introduce segmentation algorithms that address this issue.

A good performance in segmentation relies on a good representation of the features in the stack we want to segment and a good learning al-

Figure 1.2: Neural tissue image stack acquired from the Hippocampus of a rat using EM Microscopy. (a) depicts 165 slices of the stack. (b) shows the map of mitochondria in (a).

gorithm. Recent works segmenting EM images use the image grey value, Gaussian smoothing and gradient magnitude at several scales σ as basis of feature representation [90, 74] but they lack a specific selection of scales for each dataset, providing a fair feature space representation but not necessarily the best. Other approaches use the feature channels to extract integral features [26] like sums of little areas or cubes in the image [22, 5, 98]. The 3D extension of this approach does not allow to get good features in the outer part of the stack. With these approaches we are compelled either to choose between getting bad classification performance in the outer voxels of the stack or to not classify those voxels at all. In this thesis we will address this problem by describing the image at various local scales.

Convolutional neural networks (CNNs), graphical models like conditional random fields and Boosting classifiers are the most popular algorithms for neural structures segmentation. CNNs appeared only recently [7] but they need powerful graphical process units (GPU) and large amounts of training data. In this thesis we test the applicability of recent multi-class Boosting classifiers to the segmentation problem. On the other hand, Boosting algorithms, together with traditional Markov Random Fields regularization, provides a very efficient segmentation approach that can run in a standard CPU. Informative features of one structure

might not be informative or even detrimental for another structure. Thus, using a binary Boosting classifier for separating each structure from the background for segmentation is a practical solution, but not the best if the main goal is to map the entire brain image stack and its structures. To achieve a detailed construction of neuron structure maps a multi-class approach is required. In this work we introduce a multi-class segmentation algorithm based on PIBoost [29], a multi-class classifier based in Boosting with binary weak-learners, and BAdaCost [30] a cost-sensitive Boosting multi-class classifier.

It should be noted that although the focus of this work is the segmentation of electron microscope images, the algorithms introduced can be used in any other type of images. Hence, we also compare the results of our algorithms and approaches with other image datasets.

1.1 Contributions of the thesis

The following points describe the original contributions of this thesis to the image segmentation field:

- **Feature selection algorithm.** We perform a comparative study to select, among the most popular features for image segmentation, the ones that describe better the brain structures we want to segment. These features conveniently parametrized will be the pool from which the classifier learns. To this end, in chapter 3 we introduce a modification of the PIBoost algorithm [29] to perform the feature selection. We experimentally show, that by selecting the adequate set of features, it is possible to achieve better performance than the state-of-the-art approaches. Our comparative study for feature selection for brain structures segmentation was presented in [16]. The scale selection with PIBoost algorithm and its experiments was presented in [15].
- **Jaccard Curve.** The metrics used to evaluate the performance of classifiers are often reduced to a number or a percentage that represents the quality of segmentation (i.e. Jaccard index, F1-Score, etc.). The Jaccard index is a common measure for binary segmentation (see chapter 2). In this thesis we introduce a curve that presents the Jaccard values for all possible thresholds for a given segmentation algorithm. We claim that this is the right way for comparing segmentation algorithms and we use it to compare different methods in EM images of the brain. The Jaccard curve was introduced in [15].

1.1 CONTRIBUTIONS OF THE THESIS

- **Multi-class Boosting for segmentation of multiple structures.** The fundamental result in this thesis is an image segmentation algorithm specially conceived to analyse neuronal structures in EM images of the brain. To achieve this goal we had to:
 1. Use recent multi-class Boosting classification algorithms to address the segmentation problem from a multi-class perspective.
 2. Select the most discriminant image features at the right scale.
 3. Introduce a resampling scheme to address the asymmetry of the pixel classification problem.

The resulting algorithm is the state-of-the-art Boosting segmentation result. This algorithm was published in [15].

- **Optimization of segmentation using costs.** For heavily asymmetric problems with very different class prior probabilities (e.g. visual object detection problems, pixel-wise segmentation, etc.), there are performance metrics that take into account the different number of training samples within the classes: F_1 -Score, Jaccard, AUC (Area Under the ROC curve), etc. Classifiers are usually designed to minimize the classification error and after training, we need to develop an ad-hoc procedure for each family of classifiers (e.g SVM, Random Forest, etc) in order to optimize the classifier for the segmentation metric at hand. For example, with binary classifiers as AdaBoost or SVM it would imply to move the threshold using a validation set. When the problem is multi-class, changing a single threshold parameter is not an option and training a classifier to optimize the F_1 -Score, Jaccard, etc. is difficult. In this thesis we advocate for using the degrees of freedom provided by a cost-sensitive classifier, the cost matrix, as a tool to optimize a given performance metric. Costs can move all the class boundaries in order to improve the performance in a way different to minimizing the number of errors. Moreover, a cost sensitive Boosting classifier can select features different from the minimum error ones, in order to improve the segmentation measure. We show through experimentation that BAdaCost, a recently proposed multi-class cost-sensitive Boosting classifier can be optimized to significantly improve Jaccard in visual segmentation problems.

1.2 Organization of the thesis

The reminder of this thesis is presented as follows: in Chapter 2 we present the background information on Boosting-based supervised classification and segmentation, along with a brief state-of-the-art in EM segmentation. In chapter 3 we present a comparative study of the most relevant features in the literature and an algorithm for selecting the scales of the channels of those features. Chapter 4 addresses the problem of multi-class segmentation with Boosting. We present a methodology that achieves the best Boosting segmentation results, clearly outperforming the best binary approaches. In Chapter 5 we present an algorithm to segment an image optimizing image segmentation metrics, such as the Jaccard index. And finally, the conclusions of this work are presented in Chapter 6.

2

Background

Segmentation is the process of partitioning a image into multiple sets or segments that can be identified as belonging to a class or classes. Since we work with digital images those segments are sets of pixels.

Initial approaches to segmentation were region-based [78]. Consisted of thresholding the image pixel values with the goal of assigning one of two labels to each of them, depending on whether it belonged to the foreground or background. Alternative approaches focus on finding the boundaries between segments, defining these segments as the regions enclosed by those boundaries [47]. Often these intuitive and simple methods cannot solve simple real-world segmentation problems, such as for example, segmenting vehicles in unrestricted images, since they have a wide range of different shape, color, sizes and can be found in different backgrounds [91].

Non-trivial segmentation problems were rapidly found in computer vision and methods like *watershed* [101] and *split and merge* [76] were developed to attack those problems. They worked with transformations of the pixels in order to achieve a more informative version of the original image, like merging pixels and segments into larger segments starting from a very fine initial segmentation [44, 28] or tackle the problem in *pieces* of the image [44, 28].

Nevertheless, the complexity of the problems increased and now, all modern segmentation techniques make use of machine learning algorithms. The idea is to learn a model that is able to make predictions from data samples.

Machine Learning techniques for segmentation can be classified into two categories: supervised learning and unsupervised learning. Unsupervised learning tries to describe the structure of data. Some examples of un-

supervised learning are clustering with the popular algorithm *k-means* [62], that aims to partition the data samples into k clusters in which each sample belongs to the cluster with the nearest mean, or mixture of Gaussians [9], that model the feature vectors as samples from an unknown probability distribution and try to find the clusters of that distribution.

In this thesis we will be using supervised learning for classification. This is the task of learning a function that maps an input to an output based on example input-output pairs. That function is inferred from labelled training data composed of a set of training examples where each example is a pair consisting of an input object and a desired output value. A supervised learning algorithm analyses the training data and produces an inferred function, which can be used for mapping new examples. This requires the learning algorithm to generalize from the training data to unseen data samples.

2.1 Supervised learning

The goal of supervised learning is to build a model that is able to make predictions (output) over a set of data (input). In order to do so, the model needs to be trained to learn patterns in the data. Specifically, a supervised learning algorithm, takes as input data samples with known responses (predictions), and trains a model to generate reasonable predictions for new data.

Supervised learning can fit into two categories: classification and regression. The goal of regression is to predict a continuous measurement for an observation. That is, the response variables are real numbers. The goal of classification is to assign a class (or label) from a finite set of classes to an observation. That is, responses are categorical variables.

For classification, let $\mathbf{x}$ be an observation belonging to a class y . From Bayes Decision Theory we have the class-conditional density function $P(\mathbf{x}|y)$ and the known *a-priori* probability $P(y)$. Using the Bayes rule we can change the *a-priori* probability to an *a-posteriori* probability $P(y|\mathbf{x})$

$$P(y|\mathbf{x}) = \frac{P(y, \mathbf{x})}{P(\mathbf{x})} = \frac{P(\mathbf{x}|y) \cdot P(y)}{P(\mathbf{x})} \quad (2.1)$$

where $P(\mathbf{x})$ can be found by the Total Probability Rule

$$P(\mathbf{x}) = \sum_i P(\mathbf{x}|y = i) \cdot P(y = i). \quad (2.2)$$

2.1 SUPERVISED LEARNING

The decision that yields the minimum classification error is to choose the class with highest *a-posteriori* probability. Calculated with

$$\arg \max_i \frac{P(\mathbf{x}|y = i) \cdot P(y = i)}{P(\mathbf{x})}. \quad (2.3)$$

On the other hand, the risk associated with incorrect decision might be more important than the probability of error. So our decision criterion might be modified to minimize the average risk in making an incorrect decision. This risk minimization is done by adding cost to wrong classifications with a matrix $\mathbf{C}$. An element $\mathbf{C}_{i,j}$ is the cost of classifying as j a vector of class i . And the conditional risk for classifying in a class j a given instance $\mathbf{x}$ is

$$R_j(\mathbf{x}) = \sum_i \mathbf{C}_{i,j} \cdot P(y = i|\mathbf{x}) \quad (2.4)$$

and the decision of minimum risk is obtained by

$$\arg \min_j \sum_i \mathbf{C}_{i,j} \cdot P(y = i|\mathbf{x}). \quad (2.5)$$

Interestingly, the matrix with 0's in the diagonal and 1's in the rest, gives the same classification rule as in equation 2.1, a minimum error classifier. This matrix, $\mathbf{C}_{0-1}$, is the costless cost matrix. Following the framework described above, having a set of N labelled instances $\{(\mathbf{x}_i, l_i)\}, i = 1, \dots, N$; where $\mathbf{x}_i$ belongs to a domain X and l_i belongs to $L = \{1, 2, \dots, K\}$ the finite label set of the problem. $\mathcal{P}(L)$ will denote the power-set of labels, i.e. the set of all possible subsets of L we proceed to describe the first successful boosting algorithm and the boosting algorithms used in this thesis.

2.1.1 Boosting classifiers

Along this work we will be using classifiers belonging to the family of Boosting classifiers. Boosting algorithms compute an ensemble of simple classifiers called weak-learners because they are often just slightly better than random guessing. A set of many weak-learners working together make a strong-learner or a boosting classifier that can solve many classification problems.

Training a Boosting classifier is an iterative process that at each iteration trains a weak-learner, tests the performance of the weak-learner and adapts the data by giving more weight to the miss-classified samples. In the next iteration the weak-learner can focus on learning the *difficult* data (data with more weight). After enough iterations the algorithm learns the data space giving as an output a strong classifier.

AdaBoost

AdaBoost [32] was introduced in 1997 by Freund and Schapire for the problem of supervised binary classification. This popular algorithm provides a way of combining the performance of many weak-learners, $G(\mathbf{x}) : X \rightarrow L$, here $L = \{+1, -1\}$ to produce a strong classifier

$$H(\mathbf{x}) = \sum_{m=1}^M \alpha_m G_m(\mathbf{x}), \quad (2.6)$$

whose prediction is $\text{sign}(H(\mathbf{x}))$.

As said before, boosting is an iterative process that can be seen as an stage-wise algorithm fitting an *additive model* [34, 41]. At each round m , AdaBoost learns a direction for classification, $G_m(\mathbf{x}) = \pm 1$, understood as a sign on a line, and a *step size*, α_m , understood as a measure of confidence in the predictions of G_m .

Weak-learners G_m and constants α_m are estimated in such a way that they minimize a loss function [34, 86]

$$\mathcal{L}(l, H(\mathbf{x})) = \exp(-lH(\mathbf{x})) \quad (2.7)$$

defined on the value of $z = lH(\mathbf{x})$ known as *margin* [106, 1].

This is done by defining a weight distribution over the whole training set, assigning each training sample $\mathbf{x}_i$ a weight w_i . At each iteration, m , the best weak-learner is selected according to the weight distribution. This weak-learner is added to the ensemble multiplied by the goodness parameter α_m . Then, the training data $\mathbf{x}_i$ is re-weighted with $\mathcal{L}(l, \alpha_m G_m(\mathbf{x}))$. So, the weights of samples miss-classified by G_m are multiplied by e^{α_m} , and are thus increased and the weights of correctly classified samples are multiplied by $e^{-\alpha_m}$ and so decreased (see Algorithm 1). In this way, new weak-learners will focus on samples located on the class boundaries. AdaBoost is used to compare our proposals to the state-of-the-art Boosting approaches for medical image segmentation in chapter 4.

2.1 SUPERVISED LEARNING

Algorithm 1 AdaBoost algorithm.

Initialize the weight vector $\mathbf{W}$ with uniform distribution $\omega_i = 1/N, i = 1, \dots, N$.

for $m = 1$ to M **do**

Fit a classifier $G_m(\mathbf{x})$ to the training data using weights $\mathbf{W}$
 Compute weighted error: $Err_m = \sum_{i=1}^N \omega_i I(G_m(\mathbf{x}_i) \neq l_i)$
 Compute $\alpha_m = (1/2) \log((1 - Err_m) / Err_m)$
 Update weights $\omega_i \leftarrow \omega_i \cdot \exp(-\alpha_m l_i G_m(\mathbf{x}_i)), i = 1, \dots, N$.
 Re-normalize $\mathbf{W}$.

end

Output: Final classifier $\text{sign}\left(\sum_{m=1}^M \alpha_m G_m(\mathbf{x})\right)$

PIBoost

In this work we used the Partially Informative Boosting (PIBoost [29]) algorithm, a multi-class boosting classifier with binary weak-learners which facilitates the learning over multi-class weak-learners. Each PIBoost iteration fits as many weak-learners as a predefined set of label groups $E \subset \mathcal{P}(L)$ (e.g. with $L = \{1, 2, 3\}$ we could have $E = \{S_1, S_2, S_3\} = \{\{1, 2\}, \{1, 3\}, \{2, 3\}\}$).

Let K be the number of classes in the training data. For each set $S \in E$, $s = |S|$ is its cardinal. The aim of each weak-learner is to separate its associated set of labels, S , from the rest and persevere in this task iteration after iteration. A weight vector $\mathbf{W}^S$ is associated to the separator of set S . PIBoost builds a stage-wise additive model of the form $\mathbf{f}_m(\mathbf{x}) = \mathbf{f}_{m-1}(\mathbf{x}) + \beta_m \mathbf{g}_m(\mathbf{x})$. Then the algorithm estimates the goodness constant β and function $\mathbf{g}_m(\mathbf{x})$ for each label and iteration. $\mathbf{g}_m^S$ is a binary classifier that outputs a margin vector, it separates the labels in S from the rest (super-index S is omitted for ease of notation). The margin vector codifies the predicted label in a vectorial form. This codification was first introduced in [56] for extending the binary SVM to the multi-class case. Recently [106] generalizes the concept of binary margin to the multi-class case using a related vectorial codification in which a $\mathbf{J}$ -vector $\mathbf{y}$ is said to be a *margin vector* if it satisfies the sum-to-zero condition, $\mathbf{y}^\top \mathbf{1} = 0$, where $\mathbf{1}$ denotes a vector of ones. This *sum-to-zero* condition reflects the implicit nature of the response in classification problems in which each y takes one and only one value from a set of labels.

In the PIBoost case, the margin vectors (see [29]) reflect the fact that we are separating sets of labels from the rest. Lemma 2.1 solves the problem

of finding the separator $\mathbf{g}_m$ and its weight β_m , at each iteration [29].

Lemma 2.1. *Given an additive model $f_m(\mathbf{x}) = f_{m-1}(\mathbf{x}) + \beta_m \mathbf{g}_m(\mathbf{x})$ associated to a set of labels, $S \in E$, the solution to*

$$(\beta_m, \mathbf{g}_m(\mathbf{x})) = \arg \min_{\beta, \mathbf{g}(\mathbf{x})} \sum_{i=1}^N \exp \left(\frac{-\mathbf{y}_i^\top (f_{m-1}(\mathbf{x}_i) + \beta \mathbf{g}(\mathbf{x}_i))}{K} \right) \quad (2.8)$$

is

- $\mathbf{g}_m = \arg \min_{\mathbf{g}(\mathbf{x})} \sum_{i=1}^N \omega_i \cdot I(\mathbf{y}_i^\top \mathbf{g}(\mathbf{x}_i) < 0)$
- $\beta_m = s(K-s)(K-1) \log R,$

where R is the only real positive root of the polynomial

$$P_m(x) = \epsilon_1(K-s)x^{2(K-s)} + s\epsilon_2x^K - s(A_2 - \epsilon_2)x^{(K-2s)} - (K-s)(A_1 - \epsilon_1) \quad (2.9)$$

where $A_1 = \sum_{l_i \in S} \omega_i$, $A_2 = \sum_{l_i \notin S} \omega_i$, i.e. $A_1 + A_2 = 1$, $W_{m-1} = \{\omega_i\}$ the weight vector of iteration $m-1$, and $\epsilon_1 = \sum_{l_i \in S} \omega_i I(\mathbf{y}_i^\top \mathbf{g}(\mathbf{x}_i) < 0)$, $\epsilon_2 = \sum_{l_i \notin S} \omega_i I(\mathbf{y}_i^\top \mathbf{g}(\mathbf{x}_i) < 0)$

The pseudo-code of the PIBoost is in Algorithm 2. In chapter 3 we introduce a modification of PIBoost to perform feature selection in the context of image segmentation.

2.1 SUPERVISED LEARNING

Algorithm 2 PIBoost algorithm.

Initialize weight vectors $\omega_i^S = 1/N$; with $i = 1, \dots, N$ and $S \in E \subset \mathcal{P}(L)$ with $s = |S|$.

for $m = 1$ to M **do**

Fit binary classifier $T_m^S(\mathbf{x})$ over training data with respect to its corresponding ω^S . Translate $T_m^S(\mathbf{x})$ into $\mathbf{g}_m^S : X \rightarrow \hat{Y}$ with $\hat{Y}$ = the whole set of margin vectors.

Compute 2 types of errors associated with $T_m^S(\mathbf{x})$

$$\begin{aligned} \epsilon 1_{S,m} &= \sum_{l_i \in S} \omega_i^S I(l_i \notin T_m^S(\mathbf{x}_i)) \\ \epsilon 2_{S,m} &= \sum_{l_i \notin S} \omega_i^S I(l_i \notin T_m^S(\mathbf{x}_i)) \end{aligned} \quad (2.10)$$

c) Calculate R_m^S , the only real positive root of the polynomial $P_m^S(x)$ defined according to 2.9.

Calculate $\beta_m^S = s(K-s)(K-1) \log(R_m^S)$

Update weight vectors as follows:

- If $l_i \in S$ then $\omega_i^S = \omega_i^S \cdot (R_m^S)^{\pm(K-s)}$
- If $l_i \notin S$ then $\omega_i^S = \omega_i^S \cdot (R_m^S)^{\pm s}$,

where the sign depends on whether T_m^S has a failure/success on $\mathbf{x}_i$

Re-normalize weight vectors.

end

Output: Final classifier $C(\mathbf{x}) = \arg \max_k F_k(\mathbf{x})$, where $\mathbf{F}(\mathbf{x}) = (F_1(\mathbf{x}), \dots, F_K(\mathbf{x})) = \sum_{m=1}^M \sum_{S \in G} \beta_m^S \mathbf{g}_m^S(\mathbf{x})$

BAdaCost

Along with PIBoost we used the cost-sensitive algorithm BAdaCost, presented in [30] as base of our experiments in chapter 4.

BAdaCost algorithm is briefly described as follows: Let $\mathbf{C}$ be the $K \times K$ cost matrix to use in a K classes problem. Given a training sample $\{(\mathbf{x}_i, l_i)\}$ with $i = 1, \dots, N$, we minimize the empirical expected loss,

$$\sum_{i=1}^N \mathcal{L}_C(l_i, \mathbf{f}(\mathbf{x}_i)). \quad (2.11)$$

The minimization is carried out by fitting an additive model,

$$\mathbf{f}(\mathbf{x}) = \sum_{m=1}^M \beta_m \mathbf{g}_m(\mathbf{x}) \quad (2.12)$$

As in PIBoost, $\mathbf{g}_m$ is a vectorial codification of the class output by a multi-class weak-learner (see [30]). The weak learner selected at each iteration m will consists of an optimal step of size β_m along the direction $\mathbf{g}_m$ of the largest descent of the expected Cost-sensitive Multi-Class Exponential Loss Function

$$\mathcal{L}_C(l, \mathbf{g}(\mathbf{x})) = \exp(C^*(l, -) \cdot \mathbf{g}(\mathbf{x})) \quad (2.13)$$

where C^* is a $K \times K$ -matrix defined in the following way

$$C^*(i, j) = \begin{cases} C(i, j) & \text{if } i \neq j \\ -\sum_{h=1}^K C(j, h) & \text{if } i = j \end{cases}, \forall i, j \in L \quad (2.14)$$

For our cost-sensitive classification problem each value $C^*(j, j)$ will represent a "negative cost" associated to a correct classification, i.e. a "reward".

Lemma 2.2 shows how to compute the weak learner $\mathbf{g}_m$ from [30].

Lemma 2.2. β_m and $\mathbf{g}_m$ are given by minimizing:

$$(\beta_m, \mathbf{g}_m(\mathbf{x})) = \arg \min_{\beta, \mathbf{g}(\cdot)} \sum_{j=1}^K \left(S_j \exp(\beta C^*(j, j)) + \sum_{k \neq j} E_{j,k} \exp(\beta C^*(j, k)) \right), \quad (2.15)$$

where the values of $S_j = \sum_{\{n: G(\mathbf{x}_n)=l_n=j\}} w_n$, $E_{j,k} = \sum_{\{n: l_n=j, G(\mathbf{x}_n)=k\}} w_n$ and $w_n = w_n \exp(\beta_m C^*(l_n, -) \cdot \mathbf{g}_m(\mathbf{x}_n))$. Given a known direction $\mathbf{g}$, the optimal step $\beta(\mathbf{g})$ can be obtained as the solution to

$$\sum_{j=1}^K \sum_{k \neq j} E_{j,k} C^*(j, k) A(j, k)^\beta = - \sum_{j=1}^K S_j C^*(j, j) A(j, j)^\beta, \quad (2.16)$$

being $A(j, k) = \exp(C^*(j, k))$, $\forall i, j$. Finally, given a value of β , the optimal descent direction $\mathbf{g}$, equivalently $G(\cdot)$, is

$$\arg \min_{G(\cdot)} \sum_{n=1}^N w_n \left(A(l_n, l_n)^\beta I[G(\mathbf{x}_n) = l_n] + \sum_{k \neq l_n} A(l_n, k)^\beta I[G(\mathbf{x}_n) = k] \right). \quad (2.17)$$

2.1 SUPERVISED LEARNING

Algorithm 3 Cost sensitive classifier BAdaCost.

Initialize weights $\mathbf{w}$ with $w(n) = 1/N$; for $n = 1, \dots, N$

Compute matrices $\mathbf{C}^*$ with equation 2.14

for $m = 1$ to M **do**

 Obtain G_m solving 2.17 for $\beta = 1$.

 Translate G_m into $\mathbf{g}_m : X \rightarrow Y$.

 Compute $E_{j,k}$ and $S_j, \forall j, k$; as described in Lemma 1.

 Compute β_m solving equation 2.15.

$w(n) \leftarrow w(n) \exp(\beta_m \mathbf{C}^*(l_n, -) \mathbf{g}_m(\mathbf{x}_n))$.

 Re-normalize vector $\mathbf{w}$

end

Output: $H(x) = \operatorname{argmin}_k \mathbf{C}^*(k, -) (\sum_{m=1}^M \beta_m \mathbf{g}_m(\mathbf{x}))$

The BAdaCost pseudo-code is shown in Algorithm 3. At each iteration, we add a new multi-class weak learner $\mathbf{g}_m : X \rightarrow Y$ to the additive model weighted by β_m , a measure of the confidence in the prediction of $\mathbf{g}_m$. The optimal weak learner that minimizes (Eq. 2.17) is a cost-sensitive multi-class classifier trained using the data weights, w_i , and a modified cost matrix, $\mathbf{C}_{wl}$, with $C_{wl}(i, j) = A(i, j)^\beta, \forall i, j$.

2.1.2 Unbalanced vs asymmetric classification problems

Classification data often comes with high disproportion of data instances between classes. This is a very common issue in real-world image segmentation problems since they have a level of inherent asymmetry with different class priors. For example, in the segmentation of roads for automatic navigation of vehicles the road accounts for a small percentage of the data in the scene against the background (sky, side-walk, etc.). Asymmetric problems are different than unbalanced ones, since the latter, happen when there is much less training data in some classes than others in spite of all the classes having equal prior probabilities. In asymmetric problems some classes are much less probable to be found than others and thus, there is different number of data both in training and testing. (See figure 2.1).

In order to train a binary classifier with an imbalanced training set for a given performance metric, there are four common approaches [46, 64]:

1. Produce accurate estimates of the probabilities of class membership for each example and then adjust the classification threshold.
2. Use of resampling techniques in order to undersample the majority class and/or oversample the minority class.

Figure 2.1: EM image (a) with labels for structures: synapse (b) and mitochondria (c). We can see the imbalance of data belonging to class synapse against the background. In (d) we can see the noisy prediction of a classifier trying to segment mitochondria.

3. Optimize tractable measures aiming to find one that performs well for the application specific performance measure after post-processing the resulting classifier.
4. Directly optimize the application specific performance measure, which is only possible for some linear measures (e.g. imbalanced miss-classification costs with linear loss functions or ROC area).

By far, the most common procedure is to adjust the threshold to the outcome of a classifier [46, 64, 57, 75]. This method will be optimal with binary generative classifiers since we have access to the probability density functions of the classes in the new feature space regions after the threshold is changed. On the other hand, binary discriminative classifiers, like the Boosting algorithms presented in section 2.1.1, learn only the class boundaries and do not get information of the probability density functions from them.

2.1 SUPERVISED LEARNING

Among the sampling approach we can distinguish under-sampling techniques like UnderBagging [3] and RUSBoost[89] that randomly under-sample instances from the majority class in each iteration of AdaBoost.M2. An extension of RUSBoost is EUSBoost [36] that uses the evolutionary under-sampling algorithm from [37] instead of random under-sampling. Some popular techniques use the synthetic minority over-sampling algorithm [17], among them SMOTEBoost [18], MSMOTE [43] and SMOTE-Bagging [104].

The problem in sampling approaches is that they ignore the asymmetry nature of most real world problems thus having good performance in toy and curated datasets and not the same performance otherwise. Also there is a lack of optimal multi-class solutions since the good ratio of sampling or the weights of each class to be classified needs to be defined and that leads to an optimization, which is a different problem itself.

Other techniques attack the imbalance problem prioritizing difficult instances by preprocessing the data [95] or by cost-sensitive approaches [100, 97, 27].

Optimization of evaluation metrics is an open research field. According to [68], algorithms for learning to maximize F-measures follow two approaches: the empirical utility maximization (EUM) approach learns a classifier having optimal performance on training data, while the decision-theoretic approach learns a probabilistic model and then predicts labels with maximum expected F-measure.

In chapter 5, we present an algorithm to optimize Jaccard and F-measures, based on learning the weight of a cost-sensitive classification matrix with BAdaCost.

2.1.3 Segmentation metrics

We now briefly describe three of the evaluation metrics that we will be using in the rest of this thesis for comparing the results of segmentation algorithms. These metrics are widely used in the literature and give a better performance insight for segmentation tasks than simpler metrics like the accuracy.

Jaccard Index

The *Jaccard index* or Jaccard similarity coefficient measures the similarity between the ground truth and the result provided by the algorithm evaluated. It is a widely used image segmentation quality index both in the computer vision and bio-medical literature [61, 63]. It is defined as the

area of the intersection divided by the area of the union of segmentations (see Figure. 2.2). In terms of classification results it can be expressed as

$$JAC = \frac{TP}{TP + FP + FN}, \quad (2.18)$$

where TP stands for true positive, FP false positive and FN false negative. It represents a binary non-symmetric measure of coincidence of two segmentations. It takes values between 0 (no coincidence) and 1 (total coincidence). In our results Jaccard indices are computed from each positive class (e.g. mitochondria) versus the rest (e.g. background). Although this is the most usual way to show results, other works compute the average Jaccard index of positive and negative classes [61].

Figure 2.2: Jaccard similarity coefficient. (left) in blue the ground truth segmentation of a mitochondrion, in red the result obtained with an automated segmentation algorithm; TP is the intersection of red and blue regions, i.e. the correctly segmented piece of mitochondrion; FN is the only blue area, i.e. the part of mitochondrion segmented as background; FP is the only red area, i.e. the background segmented as mitochondrion; TN is the rest of the image, i.e. the correctly segmented background; (right) representation of the Jaccard coefficient as the relation between the purple and green areas.

F1-Score

A similar metric is the F1-score also known as F-score or F-measure. It considers both the precision and recall of the segmentation and is defined as the harmonic mean of the precision ($\frac{TP}{TP+FP}$) and recall ($\frac{TP}{TP+FN}$),

$$F1_{score} = 2 \cdot \frac{precision \cdot recall}{precision + recall}, \quad (2.19)$$

As in the Jaccard index, the F1-score reaches its best value at 1 (perfect precision and recall) and worst at 0.

2.2 EM IMAGE SEGMENTATION

In chapter 5 we talk about the nature of F-measures, the general formula for any F_β measure for a positive real β is

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}. \quad (2.20)$$

ROC

A receiver operating characteristic curve (ROC), or ROC Curve, is a graphical representation of the the true positive rate ($\frac{TP}{TP+FN}$) against the false positive rate ($\frac{FP}{FP+TN}$ with TN = true negative) to evaluate the performance of a binary classifier as its discrimination threshold is varied (See figure 2.3).

Figure 2.3: Example of a ROC curve over artificial data from [82]. Higher values of AUC represents better segmentation.

The ROC Curve reaches its best value when the true positive rate is equal to 1 in each threshold evaluated. In other words, better the classifications plot higher area under the curve (AUC).

2.2 EM image segmentation

We consider the problem of segmenting mitochondria and synapse that along with membranes are some of the most prominent neuronal struc-

tures (see Figure 4.1). These structures are of interest to neuroscience. The identification and quantification of distribution of synapses provides fundamental information for the study of the brain [24]. Mitochondria, on the other hand, play a key role in the cell metabolism, physiology and pathologies [14]. The accurate segmentation and restoration of neuron membranes is requisite to address the neural circuit reconstruction problem [45].

A fundamental problem when building an EM image segmentation algorithm is selecting a set of features that describe the underlying image. This is a problem that has been thoroughly studied in the literature. In [69] mitochondria of melanoma cells were segmented with textron-based features obtained by convolving training images with a filter-bank to generate filter responses. More specialized features like Radon-like [53] and Ray features [93] have shown successful results in company of other textural features since the values of the vector can not only be taken from the raw data (grey level) but also from filters applied to the data creating several channels from which the vector can be obtained. This "multi-channel" extraction is a very popular approach used in almost every work in the literature. The channels from [94], Gaussian smoothed image, gradient magnitude, Laplacian of Gaussian, Hessian eigenvalues and structure tensor eigenvalues are the most popular for the segmentation of stacks like the ones in this thesis and used in works like [5, 63, 58, 90]. Our proposal will be to apply these generic filters at several scales obtaining a bigger set of channels from which we extract features with more information like texture, edges and orientation.

One of the requirements that needs to be taken into account in a learning algorithm is the capacity to manage high dimensional feature vectors. Boosting-based classifiers can leverage with high dimensionality without incurring in prohibitive training and execution times. In [93] the Ray features are used to train an AdaBoost classifier outperforming the Haar-like features and Histograms of Oriented Gradients when applied to detecting irregularly shaped neuron nuclei and mitochondria. In [5], AdaBoost classifier is used to find context cues of synapse from a set of standard texture descriptors such as image gradients, Laplacians of Gaussians and eigenvalues of structure tensors. The context cues approach was seen before in works like [22]. To further exploit contextual information the result of extracting features at different scales usually pooled in neighbourhoods around the described voxels or in the *integral channel features* [26]. An extension of [5] is [58] where the context information is also used to segment mitochondria.

The second element in an image segmentation algorithm is a pixel clas-

2.2 EM IMAGE SEGMENTATION

sifier. Recent works show a variety of classifiers to solve medical imaging segmentation being boosting-based, graph regularization and most recently Convolutional Neural Networks (CNNs) as the most popular approaches [69]. A CNN is a deep learning technique, a sub-field of machine learning concerned with algorithms inspired by the structure and function of the brain called artificial neural networks. It is based on learning data representations, as opposed to task-specific algorithms. Learning can be supervised, semi-supervised or unsupervised and in addition to scalability, another benefit of deep learning models is their ability to perform automatic feature extraction from raw data, also called *feature learning*.

The output of these classifiers is noisy (See figure 2.1). So, typically some form of regularization is used to clean this noise. Regarding graph regularization, graph cuts [12] and conditional random fields (CRF) [54] are popular in smoothing the outputs of classifiers. They are often used as a measure to refine the results because of its ability to regularize a segmentation. This alleviate errors like voxel classified as background in the interior of a structure, noisy segmentation, and rough edges. In [60], the distances from data to support vectors of a previously trained SVM are used as features to train a Structured Support Vector Machine (SSVM) and CRF are used to regularize the output of the SSVM. In [61], the learned boundary zones of mitochondria is incorporated into the pairwise potential of a graph cut. In [74] $\alpha\beta$ -swap algorithm from [13] is used to regularize multi-class segmentation of mitochondria and synapses. In [63], a non-parametric higher-order random field is trained over the context cues algorithm from [5] to segment mitochondria.

CNNs have the state-of-the-art results in the medical image segmentation field, with works like [79] where a deep convolutional neural network approach is presented for automatic segmentation of mitochondria in brain tissue. In [19] the authors use data augmentation to alleviate limited training data and present 2D and 3D CNNs with factorized convolutions. And most recently [7] where the U-Net deep network is used in a Domain Adaptation approach for segmentation of mitochondria and synapses.

Although CNNs yield the highest performance in segmenting medical imaging [79, 19], the requirement of powerful GPUs to achieve those results put the comparison of traditional methods like boosting with CNNs in an inequitable position.

BACKGROUND

3

Channel and scale selection for EM images of the brain

Deciphering the architecture of the brain is a key challenge in science [24]. In the last few years, we have seen advances in the automated acquisition of large series of images of brain tissue [25, 50]. The analysis of these images enable the construction of detailed maps of neuron structures from which we will better understand the basic cognitive functions of the brain, such as learning, memory and its pathologies [48]. There are tools to manually analyse and segment the structures in such images [88]. However, the complexity of these images and the high number of neurons in a small section of the brain, makes the automated analysis the only practical solution.

Mitochondria and synapses are two cell structures of neurological interest that are suitable for automated processing. Synapses are the fundamental mechanism of communication between neurons. Quantification of synapses, and the identification of its types and their distribution is critical to understand how the brain works [10]. Besides providing energy to the cell, mitochondria play an important role in many essential cellular functions including signalling, differentiation, growth and death. The morphology and distribution of Mitochondria has great importance in cellular physiology [14] and synaptic function [55]. Also atypical morphologies or mitochondria distributions are indicative of abnormal cellular states or the existence of neurodegenerative diseases [20].

Recent works have proposed algorithms for synapse [52] and mitochondria segmentation [61, 38] employing various discriminating features. To extract these features some approaches use general texture operators [52], whereas others employ specifically designed measurements [61]. In this

chapter, we will study and compare the features used in these works for the problem of joint segmentation of synapses and mitochondria. In Figure 3.1 we show a slice of one of the images used in our study and its associated labels.

Figure 3.1: A patch from the stack (a) and its labels (b), class mitochondria in grey, class synapses in white.

3.1 Features

This section briefly describes the feature descriptors considered in this study. It begins with the description of simple general purpose descriptors and proceeds in order of increasing sophistication.

3.1.1 Simple Window and Histogram

A simple window based descriptor is the result of ordering and storing a vector of the $n \times n$ neighbourhood of the pixel that we want to describe. This naive descriptor has proved to be an excellent source of information for texture segmentation [99].

A histogram based descriptor takes for each pixel an $n \times n$ neighbourhood on which a grey level histogram is computed. In [61], a histogram and the *Ray* features [93] are used as elements of the feature vector for mitochondria segmentation. In this chapter, we tested the histogram and the *ray* features separately.

A scheme of these two characteristics is shown in Figure 3.2.

3.1 FEATURES

Figure 3.2: Simple window and histogram descriptors.

3.1.2 Local Binary Patterns

The local binary patterns (LBP) [77], generate a binary code with k digits taking into account for each pixel p a set of k neighbour points at an r distance, where r is the radius from the central pixel p to its neighbours. If the value of p is higher than a neighbour k_i , we insert a 0 in the binary code, or 1 if the value is lower. The feature vector is obtained from the histogram of the LBP binary codes converted to its real values in a $n \times n$ neighbourhood. This process is outlined in Figure 3.3.

Binary Code=[00110001]

Figure 3.3: Depending on the values of the neighbouring pixels, the LBP generates a binary code that can be converted to a real value. In this case the value would be: $0 \times 1 + 0 \times 2 + 1 \times 4 + 1 \times 8 + 0 \times 16 + 0 \times 32 + 0 \times 64 + 1 \times 128 = 148$.

3.1.3 GRIMS

Gaussian Rotation Invariant and Multi Scale descriptor (GRIMS) descriptors apply to each image in the stack of a set of linear Gaussian filters at differ-

ent scales to compute zero, first and second order derivatives. These linear operators are:

$$\left\{ G_\sigma *, \sigma \cdot G_\sigma * \frac{\partial}{\partial x}, \sigma \cdot G_\sigma * \frac{\partial}{\partial y}, \sigma^2 \cdot G_\sigma * \frac{\partial^2}{\partial x^2}, \right. \\ \left. \sigma^2 \cdot G_\sigma * \frac{\partial^2}{\partial xy}, \sigma^2 \cdot G_\sigma * \frac{\partial^2}{\partial y^2} \right\}, \quad (3.1)$$

where G_σ is a Gaussian filter with standard deviation σ and $*$ is the convolution operator. We will call the result of applying these operators to the image: $s_{00}, s_{10}, s_{01}, s_{20}, s_{11}$ and s_{02} , where the subscript denotes the order of the derivatives. The feature vector calculated for each voxel in the image at scale σ is $\left\{ s_{00}, \sqrt{s_{10}^2 + s_{01}^2}, \lambda_1, \lambda_2 \right\}$, where $\sqrt{s_{10}^2 + s_{01}^2}$ is the gradient magnitude and λ_1 y λ_2 are the first and second eigenvalues of the Hessian matrix. To extend this 2D feature extraction of the GRIMS to 3D the set of linear operators (zero, first and second derivatives) at scale σ ,

$$\left\{ \begin{array}{l} G_\sigma *, \sigma \cdot G_\sigma * \frac{\partial}{\partial x}, \sigma \cdot G_\sigma * \frac{\partial}{\partial y}, \sigma \cdot G_\sigma * \frac{\partial}{\partial z}, \\ \sigma^2 \cdot G_\sigma * \frac{\partial^2}{\partial x^2}, \sigma^2 \cdot G_\sigma * \frac{\partial^2}{\partial xy}, \sigma^2 \cdot G_\sigma * \frac{\partial^2}{\partial xz}, \\ \sigma^2 \cdot G_\sigma * \frac{\partial^2}{\partial y^2}, \sigma^2 \cdot G_\sigma * \frac{\partial^2}{\partial yz}, \sigma^2 \cdot G_\sigma * \frac{\partial^2}{\partial z^2} \end{array} \right\}, \quad (3.2)$$

is applied to the stack obtaining the responses $\{s_{ijk} : i + j + k \leq 2\}$, where i, j and k indicate the order of the derivatives in the X, Y and Z axes. The partial feature vector for each voxel of the stack at a given scale is

$$\left\{ s_{000}, \sqrt{s_{100}^2 + s_{010}^2 + s_{001}^2}, \lambda_1, \lambda_2, \lambda_3 \right\}, \quad (3.3)$$

where the first component is the smoothed stack in 3D, the second one is the magnitude of the gradient and λ_1, λ_2 and λ_3 are the eigenvalues of the Hessian matrix

$$\begin{pmatrix} s_{200} & s_{110} & s_{101} \\ s_{110} & s_{020} & s_{011} \\ s_{101} & s_{011} & s_{002} \end{pmatrix} \quad (3.4)$$

The complete feature vector is the concatenation of partial feature vectors at different scales $\sigma_0, \dots, \sigma_{n-1}$. Hence, if we use n scales σ , we would have $4n$ features for the 2D approach of the GRIMS and $5n$ for the 3D approach.

3.1 FEATURES

3.1.4 Ray Descriptors

Ray features [93] are used in [61] for mitochondria segmentation. They are good at extracting shape features in objects with strong edges, such as mitochondria. The features of the Ray descriptors depend on the function:

$$\mathbf{c} = c(\mathbf{I}, \mathbf{m}, \theta), \quad (3.5)$$

that computes the position $\mathbf{c}$ from the nearest edge or contour of the image $\mathbf{I}$ to the position $\mathbf{m}$ in direction defined by the angle θ . See Figure 3.4.

Figure 3.4: Function c returns the position $\mathbf{c}$ from the nearest edge or contour of the image $\mathbf{I}$ to the position $\mathbf{m}$ in direction defined by the angle θ .

The feature vector is:

$$\mathbf{f}^{\text{Ray}}(\mathbf{I}, \mathbf{m}, \theta) = [f_{\text{ndist}}, f_{\text{norm}}, f_{\text{ori}}], \quad (3.6)$$

where

$$f_{\text{ndist}}(\mathbf{I}) = \| c(\mathbf{I}, \mathbf{m}, \theta) - \mathbf{m} \| \quad (3.7)$$

$$f_{\text{norm}}(\mathbf{I}) = \| \nabla \mathbf{I}(c(\mathbf{I}, \mathbf{m}, \theta)) \| \quad (3.8)$$

$$f_{\text{ori}}(\mathbf{I}) = \frac{\nabla \mathbf{I}(c(\mathbf{I}, \mathbf{m}, \theta))}{\| \nabla \mathbf{I}(c(\mathbf{I}, \mathbf{m}, \theta)) \|} \cdot (\cos \theta, \sin \theta)^\top \quad (3.9)$$

- f_{ndist} , is the Euclidean distance to the point $\mathbf{c}$ from the point $\mathbf{m}$, where $\mathbf{c}$ is the nearest edge pixel.
- f_{norm} , returns the gradient norm in point $\mathbf{c}$.
- f_{ori} , take into account the orientation of the vector towards the nearest edge pixel. When a point $\mathbf{m}$ is near to the centre of a closed blob is evaluated, f_{ori} usually returns values close to 1.

The final step is aligning the features to a canonical orientation to make the feature vector invariant to rotations. This is important because there may be two very similar structures but having a different orientation. This is solved by sorting the vector starting with the feature that is aligned with the direction of maximum variance using the edge pixels.

3.1.5 Difference of Gaussian's

The difference of Gaussians can be utilized to increase the visibility of edges in an image and also for blob detection. This feature subtracts one blurred image from another blurred with a different scale. A Gaussian kernel suppresses only high frequency spatial information and by subtracting one image from the other, it discards all but a handful of spatial frequencies that are present in the original grey-scale image. In our approach, we used four subtractions of blurred images at different scales.

3.1.6 Laplacian Of Smoothed Image

Due to the elliptical shape of the structures we aim to segment, we test a blob detector based on the Laplacian of the Gaussian smoothed image. This is one of the first and also most common blob detectors which usually results in strong positive responses for dark blobs and strong negative responses for bright blobs of similar size. We used different scales for this approach because the output is strongly dependent on the relationship between the size of the blob structures in the image domain and the size of the Gaussian kernel used for pre-smoothing. Therefore to identify different unknown sizes of structures we used four scales which we describe later on.

3.1.7 Eigenvalues of the structure tensor

The structure tensor is a matrix derived from the gradient of a function. It represents the predominant directions of the gradient in a specified neighbourhood of a point. It is often used in image processing due to the properties of its eigenvalues λ_1, λ_2 and eigenvectors e_1, e_2 : If $\lambda_1 = \lambda_2$, the gradient in the patch has no predominant direction which indicates that the patch has rotational symmetry and if $\lambda_1 > \lambda_2$, then $\pm e_1$ is the direction that is maximally aligned with the gradient within the patch. For a image $\mathbf{I}$ the 2D structure tensor is given by

$$\begin{pmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{pmatrix},$$

3.2 SCALES SELECTION

where x and y are the variables in the image and I_x and I_y are the partial derivatives of I with respect to x and y respectively.

3.1.8 Histogram of Oriented Gradients

The histogram of oriented gradients descriptors were first described in [23] for pedestrian detection in static images. Afterwards they were tested for human detection in film and video, as well as to a variety of common animals and vehicles in static imagery. The feature vector for the HOG implementation is obtained by dividing the image into small connected regions, called cells, and for each cell compiling a histogram of gradient directions for the pixels within the cell. To build our HOG feature vector we used an 18 bins histogram representing the signed directions of the gradient, a 9 bins histogram representing the unsigned directions, 4 features representing the sum of the gradients in each direction (up, down, left, right) and the grey value of the pixel. This makes a vector with 32 features per cell.

3.1.9 Radon-Like features

It has been shown empirically that trying to segment the structures in connectome images using only geometric or textural features is not very effective. The Radon-Like features were proposed as a remedy to this problem as they are designed to leverage both the texture and the geometric information present in the connectome images to segment structures of interest. As a first step, Radon-Like features use the edge map of a image as a means to divide it into regions that are defined by the geometry of the constituent structures. Next, for each pixel, line segments with their end points on the closest edges are computed in all directions. Finally, for each pixel, a scalar value is computed along each direction using the information in the original image along these line segments using a so-called extraction function defined in [53].

3.2 Scales Selection

Properly addressing the multi-scale nature of the structures in the SBFEM images is an important issue to achieve top segmentation performance. Synapses appear in our images with various sizes and shapes. Similarly, mitochondria show up as roughly elliptical structures with very different sizes. The information provided by the features defined in the previous

section depends on the size of the image structures and the scales of the kernels used for filtering. For a given image stack, it is not clear what is the most discriminative set of GRIMS scales. An important step in our methodology is to establish them. The standard approach would optimize the segmentation performance using cross-validation over the set of scales. However, in our problem this is computationally prohibitive. To this end we introduce a new scale selection algorithm based on a generalization of the well-known AdaBoost-based greedy feature selection scheme [102] to the multi-class case. For this purpose we adapt PIBoost [29] (see Algorithm 4).

Each PIBoost iteration learns a group of weak-learners that partially solve a multi-class classification problem. Each weak-learner separates a group of classes from the rest, learned as a binary problem in which one of the groups is treated as the positive class and the rest as negative. In this context a *separator* is a classifier formed by combining the minimal set of weak-learners that solve a multi-class problem (see Figure 3.5). Each separator associates weights to training samples [29]. These weights focus the learning process on a different set of samples at each iteration thereby encouraging each weak-learner to be independent from the rest.

Figure 3.5: PIBoost *separator* in a three class problem. It is composed of three weak-learners (S_1, S_2, S_3) separating each class, e.g. C_1 :mitochondrion, C_2 :synapse, C_3 :background, from the rest.

For feature selection we modify this scheme producing a new algorithm (see Algorithm 4). The feature selection algorithm iterates over all GRIMS scales training each separator with one scale using the weighted training samples. In our case, since we consider the simultaneous segmentation of two structures, we have two positive and one negative classes, hence, each separator has associated three weak-learners (see Figure 3.5). We classify

3.3 EXPERIMENTS

the training data with each separator (GRIMS scale) and select the one with the smallest weighted error. Finally, with the error of the selected feature we update the weights of the training data according to the PI-Boost scheme [29], so that the next selected scales are independent from those selected so far. Algorithm 4 shows this process, where the actual expression of functions $\text{train}S_k(F_j)$, $\epsilon_j(F_j, S_{k,k=1\dots 3}, W_i)$ and $W(\epsilon_{\min})$ may be found in [29],

Algorithm 4 Feature selection algorithm.

Data:

F = the set of all features at all scales.

S = the set of all separators.

W = weight vector initialized with values $W_0 = 1/N$, N = number of data instances.

Result: vector with selected features $bFeat[i]$.

$error = 0$

for all iterations i do

 // In each iteration we select one feature

$\epsilon_{\min} = \infty$.

for all features $j \in F$ do

for all separators $k \in S$ do

 // In our case $k \in \{1, 2, 3\}$

 train $S_k(F_j)$ with weights W_i .

end

 compute error of F_j , $\epsilon_j(F_j, S_{k,k=1\dots 3}, W_i)$.

if $\epsilon_j < \epsilon_{\min}$ then

$bFeat[i] = F_j$.

$\epsilon_{\min} = \epsilon_j$

end

end

 update weight vector $W_{i+1} = W(\epsilon_{\min})$

end

3.3 Experiments

We used an image stack obtained from the Somatosensory cortex of a rat, with a resolution of $3.686\mu m$ per pixel. The thickness of each layer is $20\mu m$. We used 60 images of the stack for training and 10 for testing. We ran our experiments on a 16GB RAM computer with 4 cores at 3.2GHz.

The scale selection for the GRIMS and the descriptors that use a Gaussian kernel is: $\sigma_0 = 4, \sigma_1 = 5.65, \sigma_2 = 8, \sigma_3 = 11.31, \sigma_4 = 16$. In our HOG implementation we used 4×4 pixels per cell and 6×6 cells per block. Due to the block division required, we assigned the computed histogram to the central pixel in each cell.

For the histogram and simple window descriptors, we tested with several window and bin sizes. A 20×20 pixel window with 10 bins for the histogram and a 15×15 box for the simple window had the best performance.

For the LBP we used a *radius* of 10 pixels with 25 sample points from which we obtain the LBP code for each pixel. With this code, we obtain a new image stack on which we compute a 10 bins histogram on a 20×20 pixels window, from which we build our feature vector.

For classification purposes we used two algorithms. A Gaussian and a Random Forest Classifier. The Gaussian (Bayesian) classifier is a generative parametric classifier that assumes Gaussian class conditional distributions, it learns the probability distribution $P(y_i | \mathbf{x}_i)$ of the label y_i given the feature vector $\mathbf{x}$ at voxel i . With Bayes theorem we can express this distribution as the product

$$P(y_i | \mathbf{x}_i) \propto p(\mathbf{x}_i | y_i)P(y_i) \quad (3.10)$$

where $P(y_i)$ is the *a priori* probability of having the label y_i and $p(\mathbf{x}_i | y_i)$ is the probability density of the feature vector for voxels with label y_i that can be model as

$$p(\mathbf{x}_i | y_i) = \frac{1}{\sqrt{(2\pi)^k |\Sigma_y|}} e^{-\frac{1}{2}(\mathbf{x}_i - \mu_y)^T \Sigma_y^{-1} (\mathbf{x}_i - \mu_y)} \quad (3.11)$$

where μ_y and Σ_y are the mean vector and the covariance matrix of the feature vectors for voxels with label y and k is the dimension of the feature vector. We chose this classifier because of its simplicity and speed.

Random forests operate by training a multitude of decision trees and selecting the class label that is the mode of the classes output by individual trees. We used the *scikit-learn* [82] implementation of the Random Forest classifier with 100 decision trees.

The high dimensionality of the feature vectors produced by the HOG and Radon-Like feature descriptors, given the number of training samples, produce degenerate Gaussian distributions with singular covariance matrices. For this reason we do not use a Gaussian classifier with HOG and Radon-Like features in our experiments. Similarly, we do not show the results

3.3 EXPERIMENTS

Table 3.1: Learning and prediction times

Descriptor	Random Forest Classifier		Gaussian Classifier	
	Learning	Prediction	Learning	Prediction
LBP	27 min	36 s	1.4 s	1.3 s
Simple Window	126.1 min	29.4 s	3 s	1.2 s
Histogram	12.3 min	13.29 s	3 s	0.5 s
GRIMS	35.2 min	15.54 s	1.2 s	1 s
Ray's	53.1 min	19 s	2 s	0.4 s
HOG	54.4 min	21 s	-	-
Eigenvalues of structure tensor	-	-	2.1 s	0.2 s
Laplacian of Gaussian	18.4 min	11.3 s	2.3 s	1 s
Difference of Gaussians	18.6 min	11.9 s	2.3 s	1.1 s

of the Random Forest classifier with LBP features, given its poor performance.

In our experiments, we use the ROC curve of each class against the rest and the Jaccard Index as comparison indices. We have performed an extensive set of experiments involving different feature configuration. We compared our work with the Radon-Like features [53] and the work in [61] but testing the descriptors of their feature vector individually (i.e. the histogram and Ray descriptors were tested separately). For brevity, in our figures we only show the best segmentation results for the ROC curve (Figures 3.8 and 3.9) and Jaccard Index (Figure 3.7). In these figures **GC** stands for *Gaussian Classifier* and **RF** stands for *Random Forest Classifier*. In Figure 3.6 we show sample results of the best segmentation for some of the features studied. The results of our experiments show that the Random Forest classifier achieves the best performance for both mitochondria and synapse segmentation. However, as shown in Table 3.1, the time it takes to train and classify is roughly one order of magnitude larger than the Gaussian classifier preventing real-time interaction. On the other hand, the Gaussian Classifier is significantly faster than the Random Forest at expense of a marginal loss in performance. We can see this performance loss in Figure 3.8, where, for example, the mitochondria ROC Curve for the GRIMS descriptors with Random Forest is the second best result. However, at typical false positive rates between 0.05 and 0.1, the performance of a Gaussian classifier is slightly worse. Similarly, in the mitochondria plot, the ROC curves of any feature set always achieve worse

performance for the Gaussian classifier than for the Random Forest.

The GRIMS descriptors provide the best performance for mitochondria segmentation, closely followed by the simple window. On the other hand, for the segmentation of synapses, the GRIMS descriptor also provides the best performance, but at false positive rates above 0.02. The Jaccard index and the ROC curve provide similar evaluation results.

The random forest classifier also provides an importance measure that indicates the relevance for each feature. In our experiments the most discriminative feature is the grey level and we can see that the descriptors that use the grey level and its variations are the best descriptors in this work.

Although Ray descriptors [93] were specifically designed to describe structures with strong edges, such as mitochondria, they achieve their best performance when used in combination with grey value histograms [61]. Similarly, LBP's have been extensively used for describing textures, one of their features being their robustness against illumination changes. However, as our experiments suggest, they do not perform well in EM images possibly because they have a very strong background noise.

We would also like to remark on the good results achieved by the simple window descriptor. There are various reasons that justify this result. First, it is well known that texture classification using the joint distribution of intensity values over extremely compact neighbourhoods outperforms classification using sophisticated filter banks with large support [99]. Second, both mitochondria and synapses are dark regions on a clear background. Hence, they can be well represented with such a feature. Finally, this feature works slightly better for synapse than for mitochondria segmentation. The reason for this is that in our images an average synapse is around 10 pixels thick. So, in most cases it can be fully enclosed within a 15×15 window. On the other hand, mitochondria are structures much larger. Very seldom they do fit entirely within that window. Hence, they are more difficult to represent with a compact neighbourhood. However, they have a well defined texture that can be described invariantly to rotations with GRIMS. Also the difference in learning and predicting times between the GRIMS with the classifiers Random Forest and Gaussian is very significant but the difference is not as significant for performance. Results show that GRIMS and simple window descriptors exhibit the best performance being GRIMS marginally better at typical false positive rate between 0.05 and 0.1.

3.3 EXPERIMENTS

Figure 3.6: Comparative segmentation results of Random Forest classifier (RF) and Gaussian Classifier (GC) with tested features: GRIMS, simple window, histogram, difference of Gaussians (DoG), Laplacian of Gaussians (LoG) and HOG.

CHANNEL AND SCALE SELECTION FOR EM IMAGES OF THE BRAIN

(a) Random Forest Classifier

(b) Gaussian Classifier

Figure 3.7: Jaccard index of synapse and mitochondria segmentation. The top figure shows results with a random forest classifier, the bottom figure shows results with a Gaussian classifier.

(a) Mitochondria

Figure 3.8: ROC Curves. ROC Curves of class Mitochondria. Dotted lines are the experiments done with a Gaussian classifier (GC) and continuous lines with a random forest classifier (RF).

(a) Synapse

Figure 3.9: ROC Curves of class Synapse. Dotted lines are the experiments done with a Gaussian classifier (GC) and continuous lines with a random forest classifier (RF).

3.4 Conclusions

Nine feature descriptors with two classifiers in an EM stack for the joint segmentation of mitochondria and synapses were tested. Results show that GRIMS and simple window descriptors exhibit the best performance.

The GRIMS descriptors obtained the highest performance followed by the simple window descriptor both using a Random Forest classifier. The best Jaccard Indexes correspond to the simple window and GRIMS descriptors with Random Forest classifier.

We obtained the worse performance with the Ray descriptors, LBP and eigenvalues of the structure tensor descriptors. The Ray descriptors were made to work together with a histogram, giving the feature vector for mitochondria segmentation an extra tool with rotation invariance to differentiate the mitochondria from other similar structures. The LBP are efficient against illumination changes, but in our images the illumination is regulated and invariant.

Although the Random Forest classifier achieves better precision, the use of the Gaussian Classifier is suggested given the large size of the typical EM image stacks and the gains in speed provided by this classifier.

CHANNEL AND SCALE SELECTION FOR EM IMAGES OF THE BRAIN

4

Multi-class Boosting for segmentation of multiple structures

Understanding the structure, connectivity and functionality of the brain is one of the challenges faced by science in the 21st century. This grand challenge is supported by the development of multiple and complementary brain imaging modalities such as structural and functional imaging [40] and light microscopy [105, 80]. At the finest level, recent advances in SF-BEM also support this long term goal [25, 50, 66]. They have made it possible to automatically acquire long sequences of high resolution images of the brain at the nanometre scale. However, the automated interpretation of these images is still an open challenge, because of their inherent intricacy and its huge size (see Figure 4.1a).

Typical feature vectors have thousands of variables. For labelling these structures ensemble classification methods, in particular Boosting and random forests, are the most popular in EM segmentation approaches, since they can select the best subset of features on-the-fly, while training.

Random Forest and Boosting classifiers like AdaBoost and GentleBoost have been used for segmenting synapses [94, 5, 52], membranes [49] and mitochondria [93, 58, 103].

Although there are some general software tools for segmenting neuronal structures [94, 67], the best results for synapses [5, 52] and mitochondria [58, 38, 61, 63] have been achieved by algorithms specifically designed for each of them. In this chapter we study whether we can improve the performance of these approaches by simultaneously segmenting more than one structure. In particular, we will concurrently segment synapses

Figure 4.1: Hippocampus rat neural tissue SBFEM image. (a) image stack; (b) 3D reconstruction of mitochondria in the stack; (c) ground truth labels of mitochondria (blue) and synapses (red) in the first slice. Membrane labels (green) have been included only for illustrative purposes.

with mitochondria and mitochondria with membranes. Since these structures arise in SBFEM stacks with different sizes, we use the feature selection algorithm introduced in Chapter 3, section 3.2. to determine the best scales to describe them. We compare our segmentations with those that target a single structure. To this end we select three image stacks and the segmentation algorithms that have reported the best performance in each of them. To make a fair evaluation we introduce a novel quality measure tool, the *Jaccard Curve*, enabling the comparison of several segmentation

4.1 IMAGE FEATURES USED

approaches independently of the selected operational point of the classifier.

4.1 Image features used

We aim to use contextual information to label each voxel. To this end we use integral channel features based on extracting the sums over rectangular regions of a set of feature channels. The exploitation of contextual information from a scene is well established in many segmentation problems [22, 26, 5, 98]. The idea is to take features from regions in the vicinity of elements in a scene. These features can be the positions of the regions [22], distance from regions to elements [98], sum of values inside those regions [58], difference between two or more regions [26]. One of the first works implementing this approach for medical imaging is [98] where a variant of integral channel features is used. In this thesis we used a 3D approach of this way of extracting features consisting in the sum of the pixels in region with cubic form.

We obtain the channels for the integral channel feature extraction by computing a Gaussian Rotation Invariant Multi-Scale (GRIMS) descriptor and an elliptical descriptor at different scales. The elements of the selected channels can be seen in figure 4.2. We choose GRIMS because they are an excellent descriptor for segmenting mitochondria and synapses [16, 74] as seen in Chapter 3. Since vesicles are a good indicator of the existence of synapses in the vicinity (see the raw image in Figure 4.2), we also include an elliptical descriptor that provides contextual information related to the existence of vesicles.

The elliptical descriptor is the result of filtering the image with an elliptic torus-like kernel (see Figure 4.3). The shape of this kernel is controlled by the radii r_1 , r_2 and thickness w parameters. As shown in Figure 4.3, the result of convolving this kernel (left image) with a vesicle-like structure (central image) returns low values for the inner parts of the vesicle.

4.1.1 Scales selection

We used the feature selection algorithm from chapter 3. Table 4.1 shows the chosen scales for each data set. They were selected among 50 equally distributed values between 1 and 50. These are the scales used in the experiments in Section 5.2. So, our feature vectors have 20 components (4 selected scales, times 5 features per scale). Also we set the parameters of

Figure 4.2: Raw image and four image description features used in our methodology. In the raw image we highlight with blue and red color a synapse and three vesicles respectively. We can appreciate the elongated shape of the synapse and the small circular shape of the vesicles, next to the synapse.

Figure 4.3: Elliptical descriptor. (left) image showing the kernel as grey values, with both radii(r_1, r_2) and the thickness (w) parameters over-imposed; (enter) image of a vesicle-like structure; (right) response obtained when convolving the vesicle image with the elliptical descriptor kernel.

the elliptical descriptor as the average radii and width of a representative set of vesicles in the stack.

4.1 IMAGE FEATURES USED

Table 4.1: Vesicle descriptor parameters and GRIMS scales selected for each data set.

Data set	GRIMS scales	Vesicle parameters		
		r_1	r_2	w
Hippocampus	1.2, 1.6, 5.6, 6.0	5	5	2
Somatosensory cortex	1.6, 4.4, 5.6, 6.4	3	3	1
Cerebellum	1.2, 1.6, 2.0, 4.8	4	4	2

4.1.2 Feature aggregation

Once selected the best features and scales, we aggregate local evidence by computing the integral channel features on them. In our approach we use cubic regions, as shown in Figure 4.4(b).

So, the feature extraction process is as follows. For each voxel in the stack we obtain the channels by convolving it with the GRIMS and elliptical filters in a set of selected scales, $\sigma_1, \dots, \sigma_n$ (see Figure 4.4(a)). A feature associated to one voxel is the sum of the filter responses in a random neighbouring cube in a randomly selected channel (see Figure 4.4(b)).

The random selection is motivated from the nature of PIBoost as a type of ensemble learning algorithm. These algorithms, like Random Forests and AdaBoost, work by combining the prediction of a set of *base classifiers*, typically trees. A key element in the construction of these base classifiers is selecting a random subset of feature variables [42]. The introduction of this randomness in the training algorithm improves generalization. This is always the case when there are many features and most of them provide discriminatory information.

In our approach we extract 1200 features for each voxel in the stack.

MULTI-CLASS BOOSTING FOR SEGMENTATION OF MULTIPLE STRUCTURES

Figure 4.4: Feature extraction process. We apply a set of filters at scales $\sigma_1, \dots, \sigma_n$ to each stack slice. (a) two filters for one slice, s_{000} left, λ_3 right; (b) a feature of voxel V_i is the sum of the values of one box (in blue). The feature vector of V_i is the concatenation of several hundreds of such features.

4.2 Multi-class boosting with integral channel features

We also adopt a Boosting scheme to label each voxel. Our classifier, Partially Informative Boosting (PIBoost) [29], is a multi-class generalization of AdaBoost with binary weak-learners.

At the m -th iteration and for each separator S , PIBoost builds a stage-wise additive model

$$\mathbf{f}_m(\mathbf{x}) = \mathbf{f}_{m-1}(\mathbf{x}) + \beta_m^S \mathbf{g}_m^S(\mathbf{x}),$$

where $\mathbf{f}_m(\mathbf{x}) \in \mathbb{R}^c$ is the strong learner and $\mathbf{g}_m^S(\mathbf{x})$ the trained weak-learner at iteration m for separator S , β_m^S is a constant related to the accuracy of the weak-learner, and c is the number of classes in the problem. Each component in the vector $\mathbf{f}(\mathbf{x})$ represents to what extent $\mathbf{x}$ belongs to each class. $\mathbf{f}(\mathbf{x})$ satisfies the *sum-to-zero* condition, $\mathbf{f}(\mathbf{x})^\top \mathbf{1} = 0$, that guarantees that each vector takes one and only one value from the set of labels [29]. Finally, sample $\mathbf{x}$ is assigned to the class α_i associated to the maximum component of $\mathbf{f}(\mathbf{x})$

$$\mathbf{x} \in \alpha_i \Leftrightarrow i = \arg \max_j \mathbf{f}(\mathbf{x})[j],$$

where $\mathbf{f}(\mathbf{x})[j]$ denotes the j -th component of vector $\mathbf{f}(\mathbf{x})$.

4.2.1 Weighted sampling for Boosting

In the context of ensemble learning algorithms, *shrinkage* and *sub-sampling* are two ways of performing regularization in the construction of the strong classifier [42]. In the case of shrinkage, this is achieved by scaling the contribution of each base classifier by a factor $0 \leq \nu \leq 1$. In [42] it is discussed in some detail the variance-reduction mechanism of sub-sampling followed by averaging. In Boosting algorithms, in general, and in particular in PIBoost, we exploit the same sub-sampling mechanism.

To this end, we train each weak-learner with a fraction of all training data. Sub-sampling reduces training time and helps to generalize the classification. For the PIBoost experiments in Section 5.2 we use $\nu = 1$ and train each weak-learner with 10% of the data randomly sampled according to their weights (see Algorithm 5). This gives priority to instances that are hard to classify. But only by discarding data before training we still deal with millions of instances, therefore the sampling we perform is

according to the weights of the data for each separator in the boosting iterations. The sampling that we used does not affect only the majority class, hence respects the asymmetry of the problem.

Algorithm 5 Weighted Sampling algorithm.

```

1 Input: Training feature vector  $\mathbf{X}$ , labels  $Y$ 
2 Initialize the weight vector  $\mathbf{W}$  with uniform distribution  $\omega_i = 1/N, i = 1, \dots, N$ .
3 for each boosting iteration do
4     Generate  $\mathbf{x}_{train}$  by subsampling  $\mathbf{X}$  according to  $\mathbf{W}$ 
5     Generate  $y_{train}$  according to  $\mathbf{x}_{train}$ 
6     Fit a classifier  $G_m(x)$  with  $\mathbf{x}_{train}$  and  $y_{train}$ 
7     Test  $G_m(x)$  by predicting  $\mathbf{X}$ 
8     Compute measure of confidence  $\alpha$  from the prediction of  $G_m(x)$ 
9     Regularize by shrinkage:  $\alpha = \alpha \cdot \nu$ 
10    Update weights and re-normalize  $\mathbf{W}$ .
end
Output: Final strong classifier.
```

The sampling rate and weighted sample used in this chapter was motivated by the study of techniques used for very large and imbalanced data sets. Most of object detection problems face the task of segmenting small structures in a large background. The problem we are targeting in this chapter is extremely asymmetric. This big difference between the amounts of positive and negative data will bias the estimation of the classifier parameters and lead to poor generalization.

In [35] the authors present an extensive review of the state-of-the-art on ensemble techniques in the framework of imbalanced datasets. Experimental results showed that from the boosting-based ensembles, Random Undersampling Boosting [89] (RUSBoost) had the best performance. RUSBoost removes instances from the majority class by random undersampling data in each iteration and normalize the weights of the sampled data for training. This process promotes the diversity in training data, which should benefit the ensemble learning. Nevertheless we tested the RUSBoost in our experiments and obtained higher performance than RUSBoost by using our weighted sampling scheme. For this experiment we extracted a sample of 50 consecutive slices from the Hippocampus stack. We used GRIMS as features and trained a PIBoost classifier with 100 iteration. With sampling rate of 0.1 (10% of the data in each PIBoost iteration) we achieve the Jaccard indexes from table 4.2 and we compare the result of our

4.2 MULTI-CLASS BOOSTING WITH INTEGRAL CHANNEL FEATURES

scheme against RUSBoost with similar parameters, 0.1 sampling rate and 100 iterations. We can see that, for our problem, the approach presented above clearly outperforms other well established resampling techniques.

Table 4.2: Comparison of the sampling methods used in our experiments. Jaccard index results from RUSBoost and Weighted sampling approaches.

	Mitochondria	Synapse
RUSBoost	0.28	0.13
Weighted Sampling	0.42	0.35

RUSBoost showed good performance with binary synthetic imbalanced data with no more than 6000 class instances, but in this work we face datasets with more than 122 million instances of background and foreground with no more than 7 million class instances. This is, 94% of negative instances and 6% of positive instances, and since our problem is multi-class that 6% is divided in 2 different classes which gives us classes with no more than 500 thousand instances. This may not seem a very imbalanced problem since RUSBoost was tested with even more imbalanced datasets but the large amount of class instances and the random sampling makes it difficult for the classifier to learn the whole feature space. To overcome this problem the classifier needs larger number of samples in each iteration or more iterations with smaller data samples. Both options entails heavy processing or long learning time.

In a problem with high number of data, the random sampling of the majority class performed by RUSBoost makes the learning of the classifier a difficult task because it takes more learning iterations to cover the data space properly. Our weighted sample procedure entails a more greedy learning approach since it guides the learning of PIBOOST towards the data difficult to segment regardless if it is from the majority or the minority class.

4.2.2 Label regularization

The output of this classification process is noisy. We filter this noise by optimizing the energy in a Markov Random Field (MRF) with pairwise terms using the graph cut algorithm [11]. Since the standard graph cut approach is only valid for binary problems, we solve our three class regularization problem in two ways. First by setting up two one positive class-against the rest problems. Second using the $\alpha\beta$ -swap multi-class extension to graph cuts [13].

We define the weights of edges in the graph as follows. Let us denote with $\alpha_y, y \in \{\text{mitochondrion}, \text{synapse}, \text{membrane}, \text{background}\}$ each of the

class labels for a voxel. The unary term of voxel $\mathbf{x}$ for class α_j , $u(\mathbf{x}, \alpha_j)$ is given by the minus log of its posterior probability, $u(\mathbf{x}, \alpha_j) = -\log P(\alpha_j | \mathbf{f}(\mathbf{x}))$. Using the multinomial logistic expression we get

$$-\log P(\alpha_j | \mathbf{f}(\mathbf{x})) = -\log \frac{e^{\mathbf{f}(\mathbf{x})[j]}}{\sum_i e^{\mathbf{f}(\mathbf{x})[i]}} = \log \sum_i e^{\mathbf{f}(\mathbf{x})[i]} - \mathbf{f}(\mathbf{x})[j]. \quad (4.1)$$

Since $\log \sum_i e^{\mathbf{f}(\mathbf{x})[i]} \approx \max\{\mathbf{f}(\mathbf{x})\}$, then, the unary term weights are given by

$$u(\mathbf{x}, \alpha_j) = \max\{\mathbf{f}(\mathbf{x})\} - \mathbf{f}(\mathbf{x})[j]. \quad (4.2)$$

For the pair-wise terms we train a new classifier that learns the probability that a voxel belongs to a border. Here a border is a thin strip around the edge of mitochondria and synapses. This is done by setting up a PIBoost-based classifier with only two classes $\alpha_y, y \in \{border, no_border\}$. The weight of the edge connecting neighbouring voxels $\mathbf{x}$ and $\mathbf{y}$, $p(\mathbf{x}, \mathbf{y})$, is given by

$$p(\mathbf{x}, \mathbf{y}) = -\log P(\alpha_{border} | \mathbf{f}(\mathbf{x})) - \log P(\alpha_{border} | \mathbf{f}(\mathbf{y})). \quad (4.3)$$

Classification and regularization are two key stages in most segmentation algorithms. Our claim is that by considering several structures in the segmentation we improve the overall segmentation, but specifically the classification stage. The only way to check this is by separately analyzing these steps.

In our experiments we provide results with and without regularization. In [5] no regularization is done. In [63] the authors use the same regularization that we use in this thesis. And [58] uses a regularization different from the one we used. In our experiments, we leave the result with our regularization. In the text we have also included the result with their regularization. In any case, PIBoost still achieves better performance. So, the superiority of algorithms that combine the prediction of several structures still holds.

4.3 Results and Discussion

Here we describe the experiments performed to evaluate the image segmentation method described in the previous section.

4.3 RESULTS AND DISCUSSION

4.3.1 Quality measure

In binary classification problems a threshold value controls how posterior probabilities are converted into class labels. To compare the performance of two such classifiers independently of the threshold the Machine Learning community has long agreed on the use of Precision Recall (PR) or Receiver Operator Characteristic (ROC) curves instead of accuracy results [84]. Similarly, in image segmentation, simply comparing a Jaccard index may be inaccurate, since, for example, the same classification algorithm with a different classification threshold would exhibit different Jaccards.

Here we introduce the *Jaccard Curve* (JCC) as a means of comparing the performance of two segmentation algorithms independently of their classification threshold. In the horizontal axis of the JCC we represent the proportion of pixels below the positive class score threshold, i.e. the percentage of pixels in the image labelled as background. In the vertical one we plot the Jaccard of the segmentation obtained when labelling in the positive class all voxels with a score higher or equal to the threshold (see Figure 4.5). We plot the JCC by sorting all voxels according to their score and evaluating the Jaccard of the segmentations at different thresholds. The higher the JCC curve, the better the segmentation.

The evaluation of membrane segmentation using the Jaccard index has been criticized because it is commonly believed that small deviations in the detected membrane locations are acceptable, which however will cause large errors in the estimated Jaccard index. Alternative more robust metrics such as the *Rand F-score*($\mathcal{F}_r$) and *Information Theoretic F-score*($\mathcal{F}_{it}$) have been recently proposed [2]. For membrane segmentation we will also use these metrics.

4.3.2 Experiments

We use three serial section electron microscopy data sets comprising different labels, SBFEM acquisition technologies and levels of anisotropy (see Table 4.3). The first two stacks, Hippocampus and Somatosensory cortex, were acquired with FIB-SEM microscopes. They have synapses and mitochondria labels manually annotated by expert neuroanatomists¹. The Hippocampus stack has perfectly isotropic voxels with very high res-

¹The Hippocampus stack was annotated at the École Polytechnique Fédérale de Lausanne (EPFL) under the supervision of Prof. Graham Knott. The Somatosensory stack was annotated at the Cajal Cortical Circuits laboratory, Universidad Politécnica de Madrid (UPM), under the supervision of Prof. Javier de Felipe.

Figure 4.5: Example of a Jaccard curve obtained with the results of the segmentation of two synapses. We selected a hard-to-segment patch to see the segmentation improvement with different thresholds. In the vertical axis we represent the Jaccard (JAC), whereas in the horizontal the percentage of pixels segmented as background. We can see how the higher the threshold (θ) the less elements are segmented as synapse.

ution. The rat Somatosensory Cortex one has a coarser resolution with slightly an-isotropic voxels. Finally, the Cerebellum stack², was acquired with a SBF-SEM microscope. It has mitochondria and membranes labels with the largest *anisotropy factor* [74].

In a preliminary experiment we want to evaluate the quality of our PIBoost-based classifier in segmenting one specific structure. To this end we compare the segmentation results obtained when segmenting mitochondria in the Cerebellum stack with those achieved by the the Cytoseg algorithm [38]. In this case we use the evaluation metrics used in [38]. Cytoseg achieved a true positive rate (TPR) of 0.80, false positive rate (FPR) of 0.02 and an accuracy of 0.97. Due to the lack of the Jaccard evaluation metric in [38] we compare our results in the cerebellum stack following the experimentation parameters and evaluation metrics used in [38]. For mitochondria segmentation we obtained a TPR of 0.84, FPR of 0.01

²More information online in the Cell Centered Database (<http://ccdb.ucsd.edu>) with ID 8192.

4.3 RESULTS AND DISCUSSION

and accuracy of 0.98. So, for segmenting a single structure our approach provides results marginally better to those achieved by Giuli et al. [38].

Table 4.3: Data sets used in the experiments. We denote the annotated labels in each data set as synapses (Syn), mitochondria (Mit) and membranes (Mem). We select train and test sets from consecutive slices in the stack. We use approximately half of each stack for training and the rest for testing.

Dataset	Voxel size (nm)	Train (voxels)	Test (voxels)
Hippocampus (Mit & Syn)	$5 \times 5 \times 5$	$1024 \times 653 \times 165$	$1024 \times 653 \times 165$
Somatosensory (Mit & Syn)	$14.7 \times 14.7 \times 20$	$366 \times 494 \times 100$	$366 \times 494 \times 113$
Cerebellum (Mit & Mem)	$10 \times 10 \times 50$	$700 \times 700 \times 20$	$700 \times 700 \times 26$

4.3.3 Results of basic classification algorithm.

For our analysis we select the algorithms reporting the best results for each of the selected stacks. We compare our algorithm with the AdaBoost-based approach of Lucci et al. for segmenting mitochondria [60], and Becker et al. for segmenting synapses [5]. We also compare our algorithm with the Bayesian approach of Marquez et al. [74], that segments both structures. To this end, we use the code provided by the authors. For the experiments with AdaBoost we trained the algorithm with 1200 decision stumps based on *context cues* [5, 58]. For the Bayesian approach we trained a classifier with Gaussian class-conditional distributions and GRIMS features as described in [74].

Finally, for PIBoost we conducted 50 iterations training 150 decision tree weak-learners. The input to this classifier are pooled features in cubes of size $5 \times 5 \times 5$ voxels computed on the channels extracted from GRIMS and elliptical descriptors on the set of selected scales, as described in Section 5.1. In all our experiments we used the first block of consecutive slices of each stack for training, and the rest for testing (see Table 4.3).

In Figure 4.7 we show the JCC curves resulting from the segmentation of mitochondria, synapses and membranes in each of the previously described image stacks. Traditionally, segmentation results have been compared by choosing an appropriate classification threshold for the classifier and computing the Jaccard for the result. This is equivalent to selecting an operation point in the JCC. This operation point for Boosting algorithms is given by a $\text{sgn}()$ function, i.e. a zero classification threshold [42], whereas in the Bayesian classifier case it is a Maximum a Posteriori (MAP) rule [42]. In the first two columns of Table 4.4 we give the Jaccard index resulting from segmenting the image at this operation point. Moreover, in each

curve in Figure 4.7 we show with a red dot the operation point for each classifier.

In some circumstances, such as for example when classes are very imbalanced, the zero threshold of Boosting algorithms may be fine-tuned [102]. This threshold is an important parameter for reproducibility and should only be estimated on a separate validation set, never in the test set. In our analysis we do not adjust it since the JCC already provides information for all thresholds.

Figure 4.6: Membrane detail.

We introduced the Jaccard curves to evaluate the performance of a segmentation algorithm independently of the selected classification threshold. From the analysis of the Jaccard curves in Figure 4.7 we can see that, in general, the approach based on the PIBoost classification algorithm and a set of pooled GRIMS features, achieves equal or superior performance to all other approaches in all stacks and structures.

In the segmentation of both mitochondria and synapses context information plays a key role. For this reason the AdaBoost and PIBoost approaches, both based on pooled channel features, achieve the best performance on both structures in the perfectly isotropic Hippocampus stack (see Figure 4.7(a) and Table 4.4). The Somatosensory and Cerebellum stacks are increasingly anisotropic. In this case most of the close context information across slices is lost and pooled channel features become less informative. Hence, segmentation performance degrades, specially for mitochondria. However, since GRIMS channels acquired at different scales also provide some local context information, the segmentation algorithm based on PIBoost degrades to a lesser extent (see Table 4.4).

The segmentation of membranes on the most anisotropic stack (Cerebellum) (see Figure 4.7(c)) does not depend on context but on local appearance. This is due to the fact that membranes are distributed all over the stack and have very different context. In this case, the algorithms based on GRIMS, PIBoost and Bayesian approaches, provide the best per-

4.3 RESULTS AND DISCUSSION

formance in the classification stage. In Figure 4.6 we can see how AdaBoost misses most membranes in the centre of the image. When segmenting membranes, small deviations in the detected locations are acceptable, since their purpose is to separate different neurons. This, however, would cause large variations in the predicted Jaccard index. This is why the Jaccard index is not the most appropriate measure to evaluate membrane segmentation. The Jaccard curve introduced in this thesis helps, to some extent, to improve this undesirable behaviour, since it considers all possible thresholds of the positive class and, hence, would be more robust to small deviations in the membrane location, caused by an incorrect classifier threshold. This is why we have also evaluated membrane segmentation results using alternative metrics such as the *Rand F-score*($\mathcal{F}_r$) and *Information Theoretic F-score*($\mathcal{F}_{it}$), proposed in a recent international challenge [2].

The advantage of PIBoost when compared with the other approaches is qualitatively quite prominent in Figure 4.6 where we zoom the results of figure 4.8 to better appreciate the perform of PIBoost over membrane classification. If we look at the membrane around the central mitochondria, we can see that the Bayesian approach roughly reconstruct 50% of it, AdaBoost 70% and PIBoost 85%.

In Table 4.5 we show the membrane segmentation results in the Cerebellum stack using the *Rand F-score*($\mathcal{F}_r$) and the *Information Theoretic F-score*($\mathcal{F}_{it}$). Here again the approach based on combining the simultaneous segmentation of the two possible classes, PIBoost, outperforms the rest.

Its important to notice that the zero or 0.5 threshold are not necessary the ones with the optimal performance, but it is methodologically incorrect to adjust a threshold to its higher performance in the test set because this provides optimistically biased solutions. To find a good threshold it is necessary to use the train/validation set for parameter adjustment and not the testing dataset. In this work we don't adjust the thresholds, instead, we plot the JCC that provides discriminative information for all possible thresholds. Also, in the first two columns of Table 4.4 we give the Jaccard index of the *zero* operation point.

In the experiment for segmenting synapses in the Hippocampus stack using AdaBoost we kept the original parametrization of [5]. As we show in the next experiment, this provides an optimal segmentation in the central part of the stack, but, as shown in Table 4.4, very poor results elsewhere. In this stack as well as in the somatosensory one PIBoost provides the best overall performance.

Similarly, for membrane segmentation in the extremely anisotropic Cerebellum stack, PIBoost and Bayesian approaches give the best performance.

This is due both to the extremely anisotropic nature of the stack and the fact that membranes are homogeneously distributed all over it. In both situations the context features provide poor information.

4.3.4 Classification and regularization results.

After classifying each voxel we regularize the resulting labels with two standard graph-cut-based algorithms. This regularization usually boost the performance and visually improves the results for large and regular regions such as mitochondria (see Figure 4.8). However, with thin and elongated structures like synapses and membranes, graph-cut regularization can be detrimental. This may be appreciated in the regularized results for membranes in Figure 4.6.

Some approaches use a regularization scheme that has been specifically conceived for the segmentation problem addressed. This is the case, for example, of the regularization approach used for segmenting mitochondria in [58].

In the case of multi-class classifiers like PIBoost and Bayesian, our proposed regularization scheme uses both the multi-class $\alpha\beta$ -swap algorithm and, by posing it as two bi-class problems, the two-class graph-cut. The former is similar to the regularization used in [74]. For the two-class AdaBoost algorithm we use a graph-cut-based regularization. Analyzing the segmentation results before and after regularization lets us make a fair comparison with [5], that uses no regularization. However, as discussed above, for mitochondria segmentation, the approach in [58] uses a different type of regularization. With this specially conceived regularization scheme [58] achieves on the Hippocampus stack a Jaccard of 0.74, slightly better than the result with our standard graph-cut-based scheme, but still behind 0.76 achieved with PIBoost using $\alpha\beta$ -swap (see first row in Table 4.4).

4.3.5 Ablation analysis.

In the next experiment we further analyse the reason why our algorithm achieves a good segmentation accuracy. To this end we select the Hippocampus stack, for which the AdaBoost-based approach of Lucci et al. [58] is the state-of-the-art for mitochondria segmentation. The first row in Table 4.8 shows the results for the AdaBoost classifier with the integral channel features on the set of channels described in [58] with a zero-threshold classification. Only by changing the channels to GRIMS and the elliptical

Table 4.4: Quantitative segmentation results for mitochondria, synapses and membrane on the Hippocampus, Somatosensory Cortex and Cerebellum stacks evaluated with the Jaccard index.

	Segmentation		Two-class regularization		$\alpha\beta$ -swap regularization	
	Mitochondria	Synapses	Mitochondria	Synapses	Mitochondria	Synapses
PIBoost	0.70	0.36	0.73	0.41	0.76	0.34
AdaBoost [5, 58]	0.62	0.07	0.71	0.30	-	-
Bayesian [74]	0.45	0.32	0.46	0.29	0.49	0.28
Hippocampus Stack						
	Mitochondria	Synapses	Mitochondria	Synapses	Mitochondria	Synapses
PIBoost	0.54	0.35	0.57	0.41	0.57	0.31
AdaBoost [5, 58]	0.51	0.30	0.55	0.39	-	-
Bayesian [74]	0.46	0.18	0.47	0.17	0.52	0.28
Somatosensory Cortex Stack						
	Mitochondria	Membrane	Mitochondria	Membrane	Mitochondria	Membrane
PIBoost	0.52	0.36	0.57	0.31	0.56	0.31
AdaBoost [5, 58]	0.36	0.22	0.55	0.29	-	-
Bayesian [74]	0.51	0.34	0.55	0.21	0.52	0.28
Cerebellum Stack						

Table 4.5: Quantitative segmentation results for membrane Cerebellum stack evaluated with the *Rand F-score*($\mathcal{F}_r$) and *Information Theoretic F-score*($\mathcal{F}_{it}$) metrics.

	$\mathcal{F}_r$	$\mathcal{F}_{it}$
PIBoost	0.93	0.91
AdaBoost [5, 58]	0.81	0.84
Bayesian [74]	0.83	0.82
Cerebellum Stack		

Table 4.6: Results of table 4.8 excluding voxels. The performance improved by not taking into account voxels and slices in the borders in the edges of the test stack during evaluation.

Classifier	Features	Mitochondria	Synapse
AdaBoost	EPFL [5, 58]	0.721173	0.449711
AdaBoost	GRIMS + EI	0.724752	0.452947
PIBoost	GRIMS + EI	0.713690	0.390546

descriptor, we get a small improvement for mitochondria and an large improvement in synapses. This is due to the fact that the elliptical descriptor and GRIMS features provide multi-scale information fundamental for the estimation of synapses, specifically those near the borders of the stack. By changing the two-class classifier for a multi-class Boosting approach we get a new improvement in performance for both structures, as shown in the third row of Table 4.8. The final boost in performance for our approach comes from selecting the best GRIMS scales, as shown in the last row.

The results for synapses segmentation in the Hippocampus stack in Table 4.8 are worse than those in [5]. Here we analyse this discrepancy. The poor result in our experiment is caused by two factors. First and foremost, the fact that the zero threshold operation point is particularly harmful for this problem (see Figure 4.7(a)). The reason for this is that AdaBoost performs poorly on highly imbalanced classification problems, such as synapses segmentation. However, we present the results of the same segmentations of table 4.8 but excluding the edges of the stack during the evaluation in table 4.6. PIBoost, on the other hand, achieves better performance because it was conceived to address the imbalanced situations arising in multi-class classification [29]. Second, the information provided by the context cues features degrades in those voxels near the borders of the stack, since many of the cubes straddle the stack limits (Figure 4.4(b) graphically depicts this problem). This is in part alleviated by the local information provided by the GRIMS. To evaluate the impact of these issues on the results we peel off from the Jaccard index computation in the stack

4.3 RESULTS AND DISCUSSION

(a) Hippocampus Jaccard Curve

(b) Results of segmentation in the first slices of the stack.

Table 4.7: The result of excluding voxels and slices in evaluation. In (a) the performance of synapse segmentation is increased over 0.15 points in Jaccard curves obtained by excluding first 10 and last 10 slices and 10 voxels from the top and bottom of the test stack. The highest point of CCBoost Jaccard curve can be obtained by thresholding a posteriori probabilities of AdaBoost output with threshold ≈ 8 , which reach the performance presented in [5]. We can compare the curves in (a) with the equivalent curves without voxels exclusion in figure 4.7(a). A qualitative result of segmentation of 2 synapses in one of 7th slice of the stack is depicted in (b). CCBoost output is noisy and after graph cut (GC) regularization the classification improves. For PIBoost, the segmentation is less noisy. This explains the high improvement in performance of the segmentation after regularization of table 4.4 for hippocampus stack. In both PIBoost and CCBoost after regularization a entire synapse is missing.

Table 4.8: Hippocampus segmentation results (Jaccard) for different classifier, features and GRIMS scales.

Classifier	Features	Mitochondria	Synapses
AdaBoost	EPFL [5, 58]	0.618911	0.071279
AdaBoost	GRIMS+EI	0.622995	0.282041
PIBoost	GRIMS+EI	0.691545	0.347415
PiBoost	GRIMS+EI+ Scale Selection	0.705145	0.364175

the 10 voxels thick outer rind. In this case the Jaccard increases to 0.37. If we further over-fit the test data and select the best operation point, the performance goes up to 0.54, comparable with that in [5].

4.3.6 Computational cost.

Concerning the computational cost at run-time, the multi-class Boosting approach is computationally more efficient than the AdaBoost binary solution. However, the Bayesian approach is, by a large margin, the fastest algorithm. We have made these performance experiments on a computer with an Intel Xeon CPU at 2.40GHz, 96GB RAM. The PIBoost solution involves 3 separators, composed of 50 trees of depth 10. So, the classification of one voxel involves $3 \text{ separators} \times 50 \text{ trees in each separator} \times 10 \text{ inner node decisions in each tree}$. This is a total of $10 \times 50 \times 3 = 1500$ image measurements to classify one voxel. This classifier takes 90.4 minutes to label the test images in the Hippocampus stack. The AdaBoost binary solution is based on two classifiers, one for each positive class, each composed of 1200 decision stumps. Hence, to classify one voxel we have to sample $1200 \times 2 = 2400$ image values. This means that his classifier requires 128.7 minutes to label the test images in the Hippocampus stack.

The Bayesian approach trains a multi-class classifier with a feature vector of $5 \text{ features} \times 4 \text{ scales}$, $5 \times 4 = 20$ features. It uses 2.7 minutes to label the test images in the Hippocampus stack.

4.3 RESULTS AND DISCUSSION

(a) Hippocampus stack

(b) Somatosensory stack

(c) Cerebellum stack

Figure 4.7: Mitochondria, synapses and membranes JCCs for the Hippocampus (a), Somatosensory (b) and Cerebellum (c) SBFEM image stacks. In each curve we show with a red dot the zero threshold operating point for Boosting classifiers and MAP point for the Bayesian one. JCCs let us compare the segmentation performance regardless of the operating point. The higher the curve the better the segmentation algorithm.

MULTI-CLASS BOOSTING FOR SEGMENTATION OF MULTIPLE STRUCTURES

Figure 4.8: Segmentation results for each stack and algorithm after regularization.

4.4 Conclusions

In this chapter we have presented an algorithm for segmenting mitochondria, synapses and membranes in SBFEM images of brain tissue. We have shown that the segmentation accuracy in SBFEM images can be improved by simultaneously analyzing several neuronal structures. We successfully tackled this problem using PIBoost [29], a boosting algorithm for class-imbalanced problems.

We have also verified that when the set of segmented structures have different sizes, selecting a good set of scales for image description significantly improves the segmentation accuracy. To this end we use the multi-class feature selection algorithm introduced in Chapter 3, section 3.2.

Following previous results in the literature [5, 58, 74], we have also confirmed the importance of context for segmenting neuronal structures. Although pooled channels with standard image features [5, 58] provide excellent performance in the central part of an isotropic stack, we have proved that GRIMS provide better overall performance both in isotropic and anisotropic stacks due to their capacity to represent multi-scale information.

Another key ingredient of our solution is the weighted sampling scheme introduced in section 4.2.1. It not only reduces the time to train our model, but also regularizes the result and addresses the inherent asymmetry of our segmentation problem, hence improving the performance.

Considering the computational cost of the classifiers, if accuracy is the main requirement in the segmentation process, then PIBoost should be the selected classifier, since it provides the best accuracy at computational cost lower than AdaBoost. However, if computational efficiency is the main issue, then the statistical approach in [74] is, by a far margin, the fastest.

The results in this chapter are relevant to the neuroscience research community when confronting the reconstruction of the *synaptome* [24]. Firstly, because the methodology introduced is general and may be applied to segment different neuronal structures, possibly using other imaging modalities. Secondly, because when the number of neuronal structures to segment grows, if the segmentation problem is addressed one structure at a time, the computational requirements also grow, at least, linearly. However, using a simultaneous segmentation approach, the number of required features and, hence, the computational cost, increases at a slower pace, since many of these features may be shared by several structures. Moreover, since these features are selected to discriminate among a large group of structures they are more general and also achieve better

MULTI-CLASS BOOSTING FOR SEGMENTATION OF MULTIPLE STRUCTURES

segmentation accuracy, as we have confirmed in our experiments.

5

Optimizing segmentation using costs

Classification with high disproportion of data instances between classes is very common in machine learning since many real-world problems have a level of inherent asymmetry with different class priors. This asymmetry can be easily seen in detection [30] and segmentation [16, 7] problems where the background data tends to be more numerous than the foreground classes. Asymmetric problems are different than imbalanced problems, as the later classification problems happen when, in the training set there is much less data in some classes than others in spite of all the classes having equal prior probabilities. On the contrary, in asymmetric problems some classes are much less probable to be found than others and thus, there is different number of data in some classes, both in training and in testing.

There are many metrics to measure how well a classifier performs [31]. Most of the metrics used to evaluate binary asymmetric or imbalanced classification problems are non-linear performance measures based in four possible outcomes of a classifier: True Positive (TP), True Negative (TN), False Negative (FN) and False Positive (FP). Metrics like the F_1 -Score, are used commonly in classification because it favours high and balanced values of precision and recall. This performance metric gives a more insightful evaluation than the accuracy when classes are highly imbalanced and when the cost of a false positive relative to a false negative is unknown. Although those performance measures like Jaccard Index or F_1 -Score are widely used as a performance metric the classifiers are commonly designed to minimize the classification error which is a function of FN and FP. On the other hand, the use of the classification error as a measure

to train a classifier is unsuitable for problems with a heavily imbalanced training dataset as a low error can be achieved just classifying everything on the majority class [34, 32].

In binary Boosting classifiers, once the binary classifier is trained, a linear combination of weak learners builds the class boundaries that corresponds to the minimum error classification rule. If we now adjust the binary classification threshold to modify class boundaries but without computing the adequate weak learners for the new classification rule, the adjusted threshold classifier will not be optimal [65]. Moreover, we can get an optimal cost-sensitive classifier from AdaBoost by adjusting the threshold if we calibrate the probabilities in advance [75], which in fact corrects the weak learners learned with the minimum error to the new classification rule. A discriminative classifier generally does not learn the whole probability density function and in the multi-class version of this type of problem, the adjustment of class boundaries it is even more difficult to achieve after training [33].

In this chapter we present a simple and general algorithm for the optimization of the non-linear measures used to validate the performance of a classifier instead of using the classification error. We advocate to use the minimum cost classification rule (i.e. cost-sensitive classifiers) as a tool for modifying learned class boundaries in multi-class problems. There are many cost-sensitive versions for popular binary classifiers (e.g. SVMs [64] or Binary Boosting [65, 75]) but not many for multi-class classifiers.

Our procedure is similar to [73] where the learning problem is solved as an optimization problem over the set of feasible confusion matrices. We use a general optimization algorithm to find the cost matrix that yields the best performance. There are approaches to optimize the F1-Score that take advantage of the pseudo-linear nature of evaluation metrics. In [72] the authors present an adaptive linearization scheme using point-based stochastic updates as optimization for concave and pseudo-linear functions which cover several popularly used performance measures such as F-measure, G-mean and H-mean. A following work [70], builds on the framework of Narasimhan et al. (2015) [72] for unconstrained classification with complex losses, and reduces the constrained learning problem to a sequence of cost-sensitive learning tasks.

In [57], the relationship between the best achievable F-measure and the decision-making threshold that achieves this optimum value is derived from any classifier that produces a real-valued output. This can benefit other metrics like the area under the curve and Jaccard index that are also susceptible to this optimization, like in [21, 85, 71, 21] where well known approaches like Boosting, quadratic programming and support vector ma-

5.1 OPTIMIZING METRICS WITH COSTS

chines are used for optimization. And as seen recently, neural networks are also a technique that can benefit from this optimizations [6]. In [7] an approximate of the Jaccard index is used as a differentiable loss function for segmentation of medical imaging with U-nets.

Inspired by the ideas in [73, 81], we aim to find the optimum cost matrix that allows the classifier to adapt the class boundaries in order to optimize the performance metric at hand. We use BAdaCost [30], a recently proposed multi-class cost-sensitive Boosting algorithm in our experiments. In the experiments we show how, in a segmentation problem of images, we can improve the results by a large margin by training a classifier that optimizes for Jaccard Index or F1-Score. We test our approach with both metrics but in the following discussion we will be referencing to the Jaccard index to explain our approach.

5.1 Optimizing metrics with costs

In our approach we use the costs as a tool to adapt class boundaries to optimize the given performance metric. Thus, our goal is to find the cost-matrix that yields higher performance of the selected metric. To do so, we reduce the problem of F-measure maximization into a cost-sensitive classification problem but with a more general approach than [81], as ours can optimize any measure in multi-class problems.

Although we perform experiments optimizing both Jaccard index and F1-Score metrics, we will continue to explain our method for the optimization of Jaccard because the experiments in this chapter optimize the Jaccard index in order to be comparable with [63].

In order to solve the optimization problem we used the Powell's minimization algorithm [83] to find the minimum of a function without needing the derivatives. The function that we want to minimize is the negative harmonic mean of the of the multi-class Jaccard index.

$$h(\mathbf{M}) = - \left(\frac{Jaccard_1^{-1} + Jaccard_2^{-1} + \dots, Jaccard_N^{-1}}{N} \right)^{-1}, \quad (5.1)$$

where $\mathbf{M}$ is a multi-class confusion matrix and $Jaccard_i$ is the Jaccard index of the class i calculated from the one-vs-rest confusion matrix obtained from the output of a cost sensitive classifier. We used the BAdaCost [30] classifier in our experiments.

The optimization, takes as arguments the training and test sets ($\mathbf{x}_{train}, \mathbf{x}_{test}$) with its training and test labels (y_{train}, y_{test}) and a cost matrix $\mathbf{C}$. The train-

ing and test sets for optimization are obtained splitting the whole training set that will be used in segmentation.

Powell’s minimization is an iterative optimization algorithm that starts with an initial value $\mathbf{C}_0$. We make $\mathbf{C}_0 = \mathbf{C}_{0-1}$, the one corresponding to the costless multi-class classification problem. In our case $\mathbf{C}$ is a $K \times K$ matrix, where K is the number of classes in the problem. Because $\mathbf{C}$ is a cost matrix, the diagonal of $\mathbf{C}$ is 0. This mean that successfully classified samples has cost 0 and we have $(K \cdot K) - K$ parameters to optimize.

At each Powell’s minimization iteration a BAdaCost classifier is learned with a subset of the training data and a candidate cost matrix $\mathbf{C}$. The classifier is set to iterate only a few times, in our experiments we used 5 iterations which means 5 BAdaCost weak learners (our weak learners are decision trees with depth = 7). The 5 weak learners BAdacost classifier is evaluated over a validation set (different from the training samples), and the performance measure in 5.1 is computed over the predictions (i.e. using the validation set confusion matrix).

After convergence of Powell’s algorithm, we use the optimized cost matrix to train a strong BAdaCost classifier, that simply is a BAdaCost with more iterations (in our experiments 100). Algorithm 6 shows the the pseudo-code to obtain the cost matrix that minimizes 5.1.

Algorithm 6 Cost matrix optimization

Input: Cost matrix $\mathbf{C}_0$, $(\mathbf{x}_{train}, y_{train})$, $(\mathbf{x}_{test}, y_{test})$.

$\mathbf{C}^* = \mathbf{C}_0, \nu = \infty$

while *Powell has not converged* **do**

$\hat{\mathbf{C}}$ = next candidate parameters from Powell.

$\hat{H} = \text{trainBAdaCost}(\mathbf{x}_{train}, y_{train}, \hat{\mathbf{C}}, 5 \text{ iterations})$

$\hat{y} = \text{predictBAdaCost}(\hat{H}, \mathbf{x}_{test})$

$\mathbf{M} = \text{confusionMatrix}(\hat{y}, y_{test})$

$\hat{\nu} = h(\mathbf{M})$ using 5.1.

if $\hat{\nu} < \nu$ **then**

$\nu = \hat{\nu}$

$\mathbf{C}^* = \hat{\mathbf{C}}$

end

end

$H^* = \text{trainBAdaCost}(\mathbf{x}, y, \mathbf{C}^*, 100 \text{ iterations})$

Output: The optimum cost matrix $\mathbf{C}^*$, the strong classifier H^* .

With the new cost matrix $\mathbf{C}^*$ we can train a strong BAdaCost with the complete training set. Note that in algorithm 6, we can change the equa-

5.2 EXPERIMENTS AND DISCUSSION

tion 5.1 for other measures. If the new measure has a local minima near the initial parameters, the algorithm will be able to find it. The cost matrix that optimizes the performance metric will guide BAdaCost to select the best features for the task. In the experiments we have also tested a measure based on the F1-Score and the algorithm is able to get a classifier that improves the segmentation.

5.2 Experiments and Discussion

In our experiments we use the Powell’s method implemented in *python* (*scipy.optimize.minimize*). BAdaCost is implemented by us in python and the weak learners are decision trees from *scikit-learn* [82]. We use trees of depth 7 in all the experiments. And we present qualitative and quantitative results for Jaccard index optimization for MSRCv2 and Hippocampus datasets to be comparable to [63]. Nevertheless, we also optimized for F1-Score in both datasets but we only present the cost evolution of that optimization in figures 5.4 and 5.2.

5.2.1 MSRCv2 dataset

The MSRCv2 dataset [92] consists of 591 images annotated with 21 classes. The majority class has the 27.95% of all data and the minority class has the 0.87% of all data. The 591 images are split into 315 training and 276 testing images. The annotations are incomplete and with areas marked as void where none of the classes are valid. We ignore the pixels annotated as void in the ground-truth from the estimation of all validation metrics as done in [51].

For this dataset we first compute Histogram of Oriented Gradients (HOG) texture and color features extracted with 6×6 pixels per cell and 4×4 cells per patch, with 9 bins for each histogram. As done in [63] we include for each cell 4 texture descriptors and 2 color values obtaining in this way a total of $(9 + 4 + 2) \times 16 = 240$ features.

We ran our cost matrix optimization algorithm with a BAdaCost classifier with 5 iterations, shrinkage of 0.1 y and subsampling of 0.5 as described in section 10.12 of [42]. To accelerate the speed of convergence we only used 1 in each 50 pixels of the data. After convergence, we apply the cost matrix resultant to a new BAdaCost classifier with 100 iterations and the same sampling and learning rate as before. The evolution of the cost during optimization is depicted for Jaccard and F1-Score optimization in figure 5.2. Figure 5.1 shows the qualitative results of segmentation

and due to the high number of classes in this dataset we present the per-class performance of our algorithm in table 5.1 after optimizing for Jaccard index. We compare the results of our optimization with the work

Figure 5.1: Qualitative results of MSRCv2 dataset segmentations before and after our optimization.

in [63] where the authors present a novel regularization method achieving a Jaccard index of 0.236 using a random forest classifier without regularization that is similar to our 0.233487 Jaccard index achieved by BadaCost before cost optimization (using a 0 – 1 cost matrix and 100 iterations). Moreover, the results after a standard pair-wise regularization achieved 0.246 Jaccard index similar to our 0.246050 after cost matrix optimization, this is, our results without pair-wise regularizations achieves higher performances than [63] with the same features.

This dataset has a coarse and incomplete annotation of classes with significant areas of the image marked as belonging to a class where it should be marked as *void* since it does not belong to any of the classes. This areas

5.2 EXPERIMENTS AND DISCUSSION

Table 5.1: MSRCv2 dataset results before and after Jaccard index optimization.

Class	Before optimization	After optimization
background	0.353	0.355
flower	0.781	0.826
grass	0.019	0.099
sky	0.598	0.609
building	0.287	0.322
tree	0.455	0.461
water	0.349	0.359
bird	0.092	0.075
road	0.401	0.410
book	0.269	0.222
chair	0.019	0.079
cat	0.012	0.075
dog	0.094	0.075
boat	0.099	0.115
body	0.099	0.096
face	0.293	0.309
car	0.149	0.168
bicycle	0.105	0.086
cow	0.231	0.247
sheep	0.118	0.130
aeroplane	0.059	0.090

from where we obtain features can hinder the classification. To make our results comparable with [63] we ignore the pixels annotated as *void* in the ground-truth from the estimation of the Jaccard index.

From table 5.1 we can see that most of the classes with moderately homogeneous colors like *flower*, *grass*, *sky*, and *water* are the ones that yield highest performance and the ones with heterogeneous form and colors like *book*, *bird*, and *bicycle* are the ones with precarious performances. It appears that the classes in which the segmentation can be done at a pixel level (without taking into account or needing the context) yield the higher improvement after cost optimization.

From figure 5.1 we can observe that the texture described by the HOG benefit the instances of classes with constant shape as seen in the *dog* and *flower* row. Floor and walls are segmented properly along with the rest of grass. The problems are found when the color is different in areas like shadows (*flowers*) or textured elements (*dog*). This can also be seen in the glares in the class *car* and parts of a building with different colors. But

Figure 5.2: Cost evolution of the MSRCv2 data set for F1-Score and Jaccard index

in cases where the areas with same color and texture like in the *book* row, the optimization worsens the results because it tends to classify segments with similar color and texture the same most probable classes with such colors and texture when it can be part of a different class. Nonetheless, the optimization improves the performance in 16 of the 21 classes.

5.2.2 Mitochondria and synapse segmentation

For this experiment we used the electron microscopy dataset from [59], obtained from the rat hippocampus. This dataset has two stacks of images with same sizes, one stack for training and one for testing. Each stack consists of 165 slices with size 768×1024 voxels and an expert annotated in them labels three classes. The 94.32% of the 129,761,280 voxels in the test stack belongs to the class *background*, 5.30% belongs to the class *mitochondria* and 0.36% to the class *synapse*. Similar proportions can be found in the train stack.

As in Chapter 4 we compute Gaussian Rotation Invariant Multi-Scale (GRIMS) descriptor. As features for this dataset. We used $n = 4$ scales (1.2, 1.6, 5.6 and 6.0) obtained with the feature selection algorithm described in [29]. Differently to Chapter 4, we don't use sums of GRIMS in 3D cubes to get faster evaluation and less features to use.

We ran our cost matrix optimization algorithm with a BadaCost classifier with 5 iterations, sampling rate of 0.1 and learning rate of 0.01. We used 1 of each 250 voxels of the data to speed up convergence. Then we trained a BadaCost classifier with the optimized cost matrix obtained. This new classifier has the same sampling and learning rate as before and 100 iterations. The results of this experiments are shown in Table 5.2 where we

5.2 EXPERIMENTS AND DISCUSSION

improve every class with the cost matrix optimization. Figure 5.3 shows a detail of the qualitative results. In this dataset the benefits of the optim-

Table 5.2: Results of BadaCost before and after optimization for Jaccard index.

Dataset	Class	Jaccard without cost matrix optimization	Jaccard with cost matrix optimization
Hippocampus	Background	0.947154	0.963635
	Mitochondria	0.450410	0.571904
	Synapse	0.298567	0.369516

Figure 5.3: Result of segmenting mitochondria (top row) and synapses (bottom row) of a slice of Hippocampus stack before and after Jaccard index optimization.

ization are more evident than the MSRCVv2 dataset as seen in 5.2. And although the experiment is over a multi-class segmentation we present the qualitative results of figure 5.3 as a binary segmentation to appreciate the results of the Jaccard optimization more easily.

In figure 5.3 for *mitochondria* (top row) the optimization removes instances that are incorrectly classified as *mitochondria* and interestingly enough for *synapse* (bottom row), the optimization add instances of class that were incorrectly classified as something else. Moreover, the optimization seem to have smoothed the classification and due to the less number of classes and the intraclass homogeneity in texture, the convergence of the optimization is reached in less iterations than in the MSRCv2 experiments (see figure 5.4).

This experiment is not comparable with Chapter 4 or [63] since the features used for classification are not the same. In this chapter we used a

Figure 5.4: Cost evolution of the Hippocampus data set for F1-Score and Jaccard index

straightforward feature scheme in which we take the values of pixels in the channels of the GRIMS. In chapter 4 and [63] the context cues from [5] are used, giving a much higher number of possible features. This demands a higher computational requirement and time. We think that our optimization algorithm will also work with context cues features or the integral channel features that yielded better performance in Chapter 4. However, this chapter focuses in the benefits of an algorithm that optimizes a validation metric like the Jaccard index by adjusting a cost matrix associated to the problem.

5.3 Conclusions

In this chapter we presented a general algorithm that allows us to optimize useful metrics in segmentation. It seeks to minimize through Powell's method a function that is the negative harmonic mean of the per-class validation metric (see equation 5.1). Our algorithm takes the cost matrix used by our cost-sensitive classifier as variables to modify in each iteration, leading us to the best cost matrix for the problem.

We used BAdaCost as our cost-sensitive classifier to test our algorithm in the Hippocampus stack, widely used in this thesis. And we extend our work with the MSRCv2 dataset, to test out algorithm with images outside the medical imaging domain. They had satisfactory results, improving performance in almost all classes.

The results show the feasibility of the approach and an interesting line of research in segmentation with costs or in other asymmetric problems in computer vision. In a future work we plan to use the same cost based op-

5.3 CONCLUSIONS

timization with other classifiers. And we would like to apply the approach to the features in chapters [3](#) and [4](#).

OPTIMIZING SEGMENTATION USING COSTS

6

Conclusions

In the last years we have witnessed the development of imaging technologies that support the study of the brain at a the structural, functional and microscopical levels. Recent Serial Block-Face Scanning Electron Microscopes (SBFEM) can automatically acquire large sequences of nanometre-scale images of brain tissue, whose manual analysis is intractable. In this thesis we have developed novel image segmentation techniques that improve the automated analysis of these images.

We have developed an image segmentation algorithm that is the state-of-the-art Boosting scheme for segmenting mitochondria and synapses in EM images of the brain. To reach this goal we made several contributions in different areas.

We studied the most popular image feature descriptors and compared their performance for describing images produced by EM microscopes. The Gaussian Rotation Invariant Multi-scale descriptors (GRIMS) produced the best segmentation metrics. To reach this result we introduced a feature selection scheme that was able to select the best set of GRIMS scales to describe a given image stack. These results were published in [16, 39].

The comparison of different image segmentation algorithms is a challenging problem, among other reasons, because it involves matching the results of different pixel classifiers that may be tuned to different operation points. To address this problem we introduced the Jaccard Curve, as a tool to compare the performance of two segmentation algorithms independently of their operation points. We used this curve to compare the performance of our approach to the best algorithms in the literature. The Jaccard Curve was introduced in [15].

State-of-the-art EM image segmentation approaches are based on binary classifiers that segment a single image structure at a time. This not

only ignores multi-class dependencies, but also scales poorly if we want to segment the multitude of structures in the brain. To build a multi-class image segmentation scheme we use PIBoost, a recent multi-class Boosting classifier. With our approach, the number of required features and, hence, the computational cost, is smaller than in the binary case, since our features are more general and may also be shared by different structures.

Mitochondria and specially synapses take a very small portion of the input image. Segmenting such structures is a very asymmetric image classification problem. To address this we included in the Boosting scheme a resampling procedure that outperforms the state-of-the-art resampling approaches, in our EM segmentation problem.

Finally, to improve even further the quality of our results, we have also studied the problem of building classifiers that optimize an image segmentation measure, such as F1 or Jaccard indices. To this end we use BA-daCost, a recent cost-sensitive Boosting classifier, and learn the costs that optimize the desired quality index. Our experiments show very promising results, with significant improvements in the segmentations of synapses and mitochondria achieved with a pool of simple features. These results remain yet to be published.

In the very near future we would like to extend this experimentation with the state-of-the-art scale-optimized GRIMS features. We would also like to evaluate our classifiers with features obtained from a deep learning scheme and extend our analysis to other brain structures.

Semantic segmentation is one of the key problems in computer vision. In the grand scheme of things, the advances in segmentations lay the foundations on which the complete scene understanding will be constructed. With the recent popularity in deep learning techniques we have been able to see advances whose performance surpass other approaches by a large margin. We are entering an exciting age in Artificial Intelligence where the hardware in machines are keeping up with the pace of the ideas of researchers. We should work in the veracity of published methods and the accessibility of data and results to facilitate the progress in the field and to make science fully reproducible and accessible to everyone in order to take the leap that will take us to a brighter future.

6.1 Publications derived from the thesis

The comparative of feature descriptors in Chapter 3 was published in:

- Kendrick Cetina, Pablo Márquez-Neila, and Luis Baumela. "A comparative study of feature descriptors for mitochondria and synapse

6.1 PUBLICATIONS DERIVED FROM THE THESIS

segmentation." Pattern Recognition (ICPR), 2014 22nd International Conference on. IEEE, 2014.

and

- Apurba Gorai, Kendrick Cetina, Luis Baumela and A. Ghosh. "A comparative study of local binary pattern descriptors and Gabor Filter for electron microscopy image segmentation." Parallel, Distributed and Grid Computing (PDGC), 2014 International Conference on. IEEE, 2014.

The algorithm for scale selection in Chapter 3, along with the multi-class image segmentation approach described in Chapter 4 was published in:

- Kendrick Cetina, José M. Buenaposada, and Luis Baumela. "Multi-class segmentation of neuronal structures in electron microscopy images." BMC Bioinformatics 19.1 (2018): 298.

A fourth work remain yet to be published and contains the studies of Chapter 5 with the presentation of a segmentation algorithm that optimizes the F1-Score and Jaccard index.

CONCLUSIONS

Bibliography

- [1] Erin L Allwein, Robert E Schapire, and Yoram Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of machine learning research*, 1(Dec):113–141, 2000. [22](#)
- [2] Ignacio Arganda-Carreras, Srinivas C. Turaga, Daniel R. Berger, Dan Cireşan, Alessandro Giusti, Luca M. Gambardella, Jürgen Schmidhuber, Dmitry Laptev, Sarvesh Dwivedi, Joachim M. Buhmann, Ting Liu, Mojtaba Seyedhosseini, Tolga Tasdizen, Lee Kamentsky, Radim Burget, Vaclav Uher, Xiao Tan, Changming Sun, Tuan D. Pham, Erhan Bas, Mustafa G. Uzunbas, Albert Cardona, Johannes Schindelin, and H. Sebastian Seung. Crowdsourcing the creation of image segmentation algorithms for connectomics. *Frontiers in Neuroanatomy*, 9:142, 2015. [63](#), [67](#)
- [3] Ricardo Barandela, Rosa Maria Valdovinos, and José Salvador Sánchez. New applications of ensembles of classifiers. *Pattern Analysis & Applications*, 6(3):245–256, 2003. [29](#)
- [4] Lauren Barghout and Lawrence Lee. Perceptual information processing system, March 25 2004. US Patent App. 10/618,543. [13](#)
- [5] Carlos Becker, Karim Ali, Graham Knott, and Pascal Fua. Learning context cues for synapse segmentation. *IEEE Transactions on Medical Imaging*, 31(2):474–486, 2012. [12](#), [15](#), [32](#), [33](#), [53](#), [55](#), [62](#), [65](#), [67](#), [68](#), [69](#), [70](#), [71](#), [72](#), [75](#), [86](#)
- [6] Maxim Berman, Amal Rannen Ep Triki, and Matthew Blaschko. The lovász-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks. 2018. [79](#)
- [7] Róger Bermúdez-Chacón, Pablo Márquez-Neila, Mathieu Salzmann, and Pascal Fua. A domain-adaptive two-stream u-net for electron microscopy image segmentation. In *Biomedical Imaging (ISBI 2018), 2018 IEEE 15th International Symposium on*, pages 400–404. IEEE, 2018. [15](#), [33](#), [77](#), [79](#)
- [8] Serge Beucher and Michel Bilodeau. Road segmentation and obstacle detection by a fast watershed transformation. In *Intelligent Vehicles' 94 Symposium, Proceedings of the*, pages 296–301. IEEE, 1994. [13](#)

BIBLIOGRAPHY

- [9] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 1 edition, 2007. [20](#)
- [10] Lidia Blazquez-Llorca, Ángel Merchán-Pérez, José Rodrigo Rodríguez, Jorge Gascón, and Javier DeFelipe. FIB/SEM technology and alzheimer's disease: Three-dimensional analysis of human cortical synapses. *Journal of Alzheimer's Disease*, 34(4):995–1013, 2013. [14](#), [35](#)
- [11] Yuri Boykov and Gareth Funka-Lea. Graph cuts and efficient N-D image segmentation. *International Journal of Computer Vision*, 70(2):109–131, 2006. [61](#)
- [12] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, 2004. [33](#)
- [13] Yuri Boykov, Veksler Olga, and Zabih Ramin. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 23(11):1222–1239, November 2001. [33](#), [61](#)
- [14] Silvia Campello and Luca Scorrano. Mitochondrial shape changes: orchestrating cell pathophysiology. *EMBO Reports*, 11(9):678–684, 2010. [14](#), [32](#), [35](#)
- [15] Kendrick Cetina, José M Buenaposada, and Luis Baumela. Multi-class segmentation of neuronal structures in electron microscopy images. *BMC bioinformatics*, 19(1):298, 2018. [16](#), [17](#), [89](#)
- [16] Kendrick Cetina, Pablo Márquez-Neila, and Luis Baumela. A comparative study of feature descriptors for mitochondria and synapse segmentation. In *International Conference on Pattern Recognition (ICPR)*, pages 3215–3220, 2014. [16](#), [55](#), [77](#), [89](#)
- [17] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002. [29](#)
- [18] Nitesh V Chawla, Aleksandar Lazarevic, Lawrence O Hall, and Kevin W Bowyer. Smoteboost: Improving prediction of the minority class in boosting. In *Knowledge Discovery in Databases: PKDD 2003*, pages 107–119. Springer, 2003. [29](#)
- [19] Hsueh-Chien Cheng and Amitabh Varshney. Volume segmentation using convolutional neural networks with limited training data. In *Image Processing (ICIP), 2017 IEEE International Conference on*, pages 590–594. IEEE, 2017. [33](#)

BIBLIOGRAPHY

- [20] Dong-Hyung Cho, Tomohiro Nakamura, and Stuart A. Lipton. Mitochondrial dynamics in cell death and neurodegeneration. *Cellular and Molecular Life Sciences*, 67(20):3435–3447, 2010. [14](#), [35](#)
- [21] Corinna Cortes and Mehryar Mohri. AUC optimization vs. error rate minimization. In *Advances in neural information processing systems*, pages 313–320. MIT Press, 2004. [78](#)
- [22] Antonio Criminisi, Duncan Robertson, Ender Konukoglu, Jamie Shotton, Sayan Pathak, Steve White, and Khan Siddiqui. Regression forests for efficient anatomy detection and localization in computed tomography scans. *Medical image analysis*, 17(8):1293–1303, 2013. [15](#), [32](#), [55](#)
- [23] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 886–893, 2005. [41](#)
- [24] Javier DeFelipe. From the connectome to the synaptome: An epic love story. *Science*, 330(6008):1198–1201, 2010. [32](#), [35](#), [75](#)
- [25] Winfried Denk and Heinz Horstmann. Serial block-face scanning electron microscopy to reconstruct three-dimensional tissue nanostructure. *PLoS biology*, 2(11):e329, November 2004. [14](#), [35](#), [53](#)
- [26] Piotr Dollár, Zhuowen Tu, Pietro Perona, and Serge Belongie. Integral channel features. In *Proceedings British Machine Vision Conference*, 2009. [15](#), [32](#), [55](#)
- [27] Wei Fan, Salvatore J Stolfo, Junxin Zhang, and Philip K Chan. Adacost: misclassification cost-sensitive boosting. In *ICML*, volume 99, pages 97–105. Citeseer, 1999. [29](#)
- [28] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision (IJCV)*, 59(2):167–181, September 2004. [19](#)
- [29] Antonio Fernández-Baldera and Luis Baumela. Multi-class boosting with asymmetric binary weak-learners. *Pattern Recognition*, 47(5):2080–2090, 2014. [16](#), [23](#), [24](#), [42](#), [43](#), [59](#), [70](#), [75](#), [84](#)
- [30] Antonio Fernández-Baldera, José M. Buenaposada, and Luis Baumela. Badacost: Multi-class boosting with costs. *Pattern Recognition*, 79:467 – 479, 2018. [16](#), [25](#), [26](#), [77](#), [79](#)
- [31] César Ferri, Hernández-Orallo José, and Modroiu R. An experimental comparison of performance measures for classification. *Pattern Recognition Letters*, 30(1):27 – 38, 2009. [77](#)

BIBLIOGRAPHY

- [32] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997. [22](#), [78](#)
- [33] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, NY, USA:, 2001. [78](#)
- [34] Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics*, 28(2):337–407, 2000. [22](#), [78](#)
- [35] Mikel Galar, Alberto Fernandez, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 42(4):463–484, 2012. [60](#)
- [36] Mikel Galar, Alberto Fernández, Edurne Barrenechea, and Francisco Herrera. Eusboost: Enhancing ensembles for highly imbalanced data-sets by evolutionary undersampling. *Pattern Recognition*, 46(12):3460–3471, 2013. [29](#)
- [37] Salvador García and Francisco Herrera. Evolutionary undersampling for classification with imbalanced datasets: Proposals and taxonomy. *Evolutionary computation*, 17(3):275–306, 2009. [29](#)
- [38] Richard Giuly, Maryann Martone, and Mark Ellisman. Method: automatic segmentation of mitochondria utilizing patch classification, contour pair classification, and automatically seeded level sets. *BMC Bioinformatics*, 13(1):29, 2012. [35](#), [53](#), [64](#), [65](#)
- [39] Apurba Gorai, Kendrick Cetina, Luis Baumela, and Ashish Ghosh. A comparative study of local binary pattern descriptors and gabor filter for electron microscopy image segmentation. In *International Conference on Parallel, Distributed and Grid Computing (PDGC)*, pages 76–81, 2014. [89](#)
- [40] Evian Gordon. Brain imaging technologies: How, what, when and why? *Australian & New Zealand Journal of Psychiatry*, 33(2):187–196, 1999. [53](#)
- [41] Trevor J Hastie and Robert J Tibshirani. Generalized additive models, volume 43 of monographs on statistics and applied probability, 1990. [22](#)
- [42] Trevor J. Hastie, Robert John Tibshirani, and Jerome H. Friedman. *The elements of statistical learning : data mining, inference, and prediction*. Springer, 2009. [57](#), [59](#), [65](#), [81](#)

BIBLIOGRAPHY

- [43] Shengguo Hu, Yanfeng Liang, Lintao Ma, and Ying He. Msmote: improving classification performance when training data is imbalanced. In *2009 Second International Workshop on Computer Science and Engineering*, pages 13–17. IEEE, 2009. [29](#)
- [44] Anil Jain, Alexander Topchy, Martin Law, and Joachim Buhmann. Landscape of clustering algorithms. In *Proc. International Conference on Pattern Recognition (ICPR)*, volume 1, pages 260–263, Aug 2004. [19](#)
- [45] Viren Jain, Joseph F Murray, Fabian Roth, Srinivas Turaga, Valentin Zhigulin, Kevin L Briggman, Moritz N Helmstaedter, Winfried Denk, and H Sebastian Seung. Supervised learning of image restoration with convolutional networks. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007. [32](#)
- [46] Thorsten Joachims. A support vector method for multivariate performance measures. In *Proceedings of the 22Nd International Conference on Machine Learning, ICML '05*, pages 377–384, 2005. [27](#), [28](#)
- [47] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision (IJCV)*, 1(4):321–331, 1988. [19](#)
- [48] Narayanan Kasthuri and Jeff W Lichtman. Neurocartography. *Neuropsychopharmacology*, 35(1):342–343, 2010. [35](#)
- [49] Verena Kaynig, Thomas Fuchs, and Joachim M Buhmann. Neuron geometry extraction by perceptual grouping in ssTEM images. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2902–2909, 2010. [53](#)
- [50] Graham Knott, Herschel Marchman, David Wall, and Ben Lich. Serial section scanning electron microscopy of adult brain tissue using focused ion beam milling. *Journal of Neuroscience*, 28(12):2959–2964, March 2008. [14](#), [35](#), [53](#)
- [51] Peter Kotschieder, Samuel Rota Buló, Horst Bischof, and Marcello Pelillo. Structured class-labels in random forests for semantic image labelling. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2190–2197. IEEE, Nov 2011. [81](#)
- [52] Anna Kreshuk, Christoph N Straehle, Christoph Sommer, Ullrich Koethe, Marco Cantoni, Graham Knott, and Fred A Hamprecht. Automated detection and segmentation of synaptic contacts in nearly isotropic serial electron microscopy images. *PloS one*, 6(10):e24899–e24899, 2011. [35](#), [53](#)

BIBLIOGRAPHY

- [53] Ritwik Kumar, Amelio Vázquez-Reina, and Hanspeter Pfister. Radon-like features and their application to connectomics. In *Proc. Int. Conference on Computer Vision and Pattern Recognition Workshops*, pages 186–193, 2010. [32](#), [41](#), [45](#)
- [54] John D. Lafferty, Andrew McCallum, and Fernando C.N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. pages 282–289, 2001. [33](#)
- [55] Doyun Lee, Kyu-Hee Lee, Won-Kyung Ho, and Suk-Ho Lee. Target cell-specific involvement of presynaptic mitochondria in post-tetanic potentiation at hippocampal mossy fiber synapses. *The Journal of Neuroscience*, 27(50):13603–13613, 2007. [35](#)
- [56] Yoonkyung Lee, Yi Lin, and Grace Wahba. Multicategory support vector machines: Theory and application to the classification of microarray data and satellite radiance data. *Journal of the American Statistical Association*, 99(465):67–81, 2004. [23](#)
- [57] Zachary C Lipton, Charles Elkan, and Balakrishnan Naryanaswamy. Optimal thresholding of classifiers to maximize f1 measure. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 225–239. Springer, 2014. [28](#), [78](#)
- [58] Aurélien Lucchi, Carlos Becker, Pablo Márquez Neila, and Pascal Fua. Exploiting enclosing membranes and contextual cues for mitochondria segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 65–72, 2014. [32](#), [53](#), [55](#), [62](#), [65](#), [68](#), [69](#), [70](#), [72](#), [75](#)
- [59] Aurélien Lucchi, Yunpeng Li, and Pascal Fua. Learning for structured prediction using approximate subgradient descent with working sets. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1987–1994, 2013. [84](#)
- [60] Aurélien Lucchi, Pablo Márquez-Neila, Carlos Becker, Yunpeng Li, Kevin Smith, Graham Knott, and Pascal Fua. Learning structured models for segmentation of 2-d and 3-d imagery. *Transactions on Medical Imaging*, 34(5):1096–1110, 2015. [33](#), [65](#)
- [61] Aurélien Lucchi, Kevin Smith, Radhakrishna Achanta, Graham Knott, and Pascal Fua. Supervoxel-based segmentation of mitochondria in em image stacks with learned shape features. *IEEE Transactions on Medical Imaging*, 31(2):474–486, 2012. [29](#), [30](#), [33](#), [35](#), [36](#), [39](#), [45](#), [46](#), [53](#)
- [62] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on*

BIBLIOGRAPHY

- mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967. [20](#)
- [63] Pablo Márquez-Neila, Pushmeet Kohli, Carsten Rother, and Luis Baumela. Non-parametric higher-order random fields for image segmentation. In *European Conference on Computer Vision (ECCV)*, volume 8694 of *LNCS*, pages 269–284. Springer, 2014. [13](#), [29](#), [32](#), [33](#), [53](#), [62](#), [79](#), [81](#), [82](#), [83](#), [85](#), [86](#)
- [64] Hamed Masnadi-Shirazi and Nuno Vasconcelos. Risk minimization, probability elicitation, and cost-sensitive svms. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, June 21-24, 2010, Haifa, Israel, pages 759–766, 2010. [27](#), [28](#), [78](#)
- [65] Hamed Masnadi-Shirazi and Nuno Vasconcelos. Cost-sensitive boosting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(2):294–309, 2011. [78](#)
- [66] Ángel Merchán-Pérez, José-Rodrigo Rodríguez, Lidia AlonsoNanclares, Andreas Schertel, and Javier DeFelipe. Counting synapses using FIB/SEM microscopy: a true revolution for ultrastructural volume reconstruction. *Frontiers in Neuroanatomy*, 3:18, 2009. [53](#)
- [67] Juan Morales, Lidia Alonso-Nanclares, José-Rodrigo Rodríguez, Javier DeFelipe, Ángel Rodríguez, and Ángel Merchán-Pérez. Espina: a tool for the automated segmentation and counting of synapses in large stacks of electron microscopy images. *Frontiers in Neuroanatomy*, 5, 2011. [53](#)
- [68] Ye Nan, Kian Ming Chai, Wee Sun Lee, and Hai Leong Chieu. Optimizing f-measure: A tale of two approaches. *arXiv preprint arXiv:1206.4625*, 2012. [29](#)
- [69] Rajesh Narasimha, Hua Ouyang, Alexander Gray, Steven W. McLaughlin, and Sriram Subramaniam. Automatic joint classification and segmentation of whole cell 3d images. *Pattern Recognition*, 42(6):1067 – 1079, 2009. [32](#), [33](#)
- [70] Harikrishna Narasimhan. Learning with complex loss functions and constraints. In *International Conference on Artificial Intelligence and Statistics*, pages 1646–1654, 2018. [78](#)
- [71] Harikrishna Narasimhan and Shivani Agarwal. Support vector algorithms for optimizing the partial area under the roc curve. *Neural computation*, 29(7):1919–1963, 2017. [78](#)
- [72] Harikrishna Narasimhan, Purushottam Kar, and Prateek Jain. Optimizing non-decomposable performance measures: A tale of two classes. In *International Conference on Machine Learning*, pages 199–208, 2015. [78](#)

BIBLIOGRAPHY

- [73] Harikrishna Narasimhan, Harish Ramaswamy, Aadirupa Saha, and Shivani Agarwal. Consistent multiclass algorithms for complex performance measures. In *International Conference on Machine Learning*, pages 2398–2407, 2015. [78](#), [79](#)
- [74] Pablo Márquez Neila, Luis Baumela, Juncal González-Soriano, José-Rodrigo Rodríguez, Javier DeFelipe, and Ángel Merchán-Pérez. A fast method for the segmentation of synaptic junctions and mitochondria in serial electron microscopic images of the brain. *Neuroinformatics*, 14(2):235–250, 2016. [15](#), [33](#), [55](#), [64](#), [65](#), [68](#), [69](#), [70](#), [75](#)
- [75] Nikolaos Nikolaou, Narayanan Edakunni, Meelis Kull, Peter Flach, and Gavin Brown. Cost-sensitive boosting algorithms: Do we really need them? *Machine Learning*, 104(2):359–384, Sep 2016. [28](#), [78](#)
- [76] Ron Ohlander, Keith Price, and D. Raj Reddy. Picture segmentation using a recursive region splitting method. *Computer Graphics and Image Processing*, 8(3):313 – 333, 1978. [19](#)
- [77] Timo Ojala, Matti Pietikäinen, and David Harwood. A comparative study of texture measures with classification based on featured distributions. *Pattern recognition*, 29(1):51–59, 1996. [37](#)
- [78] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics*, 9(1):62–66, 1979. [19](#)
- [79] Ismail Oztel, Gozde Yolcu, Ilker Ersoy, Tommi White, and Filiz Bunyak. Mitochondria segmentation in electron microscopy volumes using deep convolutional neural network. In *Bioinformatics and Biomedicine (BIBM), 2017 IEEE International Conference on*, pages 1195–1200. IEEE, 2017. [33](#)
- [80] Jincheng Pang, Nurdan Özkucur, Michael Ren, David L. Kaplan, Michael Levin, and Eric L. Miller. Automatic neuron segmentation and neural network analysis method for phase contrast microscopy images. *Biomed. Opt. Express*, 6(11):4395–4416, Nov 2015. [53](#)
- [81] Shameem Puthiya Parambath, Nicolas Usunier, and Yves Grandvalet. Optimizing f-measures by cost-sensitive classification. In *Advances in Neural Information Processing Systems*, pages 2123–2131, 2014. [79](#)
- [82] Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Vincent Weiss, Ron an Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. [7](#), [31](#), [44](#), [81](#)

BIBLIOGRAPHY

- [83] Michael JD Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The computer journal*, 7(2):155–162, 1964. [79](#)
- [84] Foster J. Provost, Tom Fawcett, and Ron Kohavi. The case against accuracy estimation for comparing induction algorithms. In *International Conference on Machine Learning*, pages 445–453, 1998. [63](#)
- [85] Alain Rakotomamonjy. Optimizing area under roc curve with svms. In *ROCAI*, pages 71–80, 2004. [78](#)
- [86] Robert E Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. *Machine learning*, 37(3):297–336, 1999. [22](#)
- [87] Alexander Schick, Martin Bäumel, and Rainer Stiefelhagen. Improving foreground segmentations with probabilistic superpixel markov random fields. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*, pages 27–31. IEEE, 2012. [13](#)
- [88] Benjamin Schmid, Johannes Schindelin, Albert Cardona, Mark Longair, and Martin Heisenberg. A high-level 3D visualization api for java and imagej. *BMC Bioinformatics*, 11(274):274, 2010. [35](#)
- [89] Chris Seiffert, Taghi M Khoshgoftaar, Jason Van Hulse, and Amri Napolitano. Rusboost: A hybrid approach to alleviating class imbalance. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 40(1):185–197, 2010. [29](#), [60](#)
- [90] Mojtaba Seyedhosseini and Tolga Tasdizen. Multi-class multi-scale series contextual model for image segmentation. *Transactions on Image Processing*, 22(11):4486–4496, 2013. [15](#), [32](#)
- [91] Mehmet Sezgin and Bülent Sankur. Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic imaging*, 13(1):146–166, 2004. [19](#)
- [92] Jamie Shotton, John Winn, Carsten Rother, and Antonio Criminisi. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *European conference on computer vision*, pages 1–15. Springer, 2006. [81](#)
- [93] Kevin Smith, Alan Carleton, and Vincent Lepetit. Fast ray features for learning irregular shapes. In *Proc. Int. Conference on Computer Vision, ICCV*, pages 397–404, 2009. [32](#), [36](#), [39](#), [46](#), [53](#)
- [94] Christoph Sommer, Christoph Straehle, U Kothe, and Fred A Hamprecht. Ilastik: Interactive learning and segmentation toolkit. In *Biomedical Imaging*:

BIBLIOGRAPHY

- From Nano to Macro, 2011 IEEE International Symposium on*, pages 230–233. IEEE, 2011. [32](#), [53](#)
- [95] Jerzy Stefanowski and Szymon Wilk. Selective pre-processing of imbalanced data for improving classification performance. In *Data Warehousing and Knowledge Discovery*, pages 283–292. Springer, 2008. [29](#)
- [96] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010. [13](#)
- [97] Yuchun Tang, Yan-Qing Zhang, Nitesh V Chawla, and Sven Krasser. Svms modeling for highly imbalanced classification. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 39(1):281–288, 2009. [29](#)
- [98] Zhuowen Tu, Katherine L Narr, Piotr Dollár, Ivo Dinov, Paul M Thompson, and Arthur W Toga. Brain anatomical structure segmentation by hybrid discriminative/generative models. *Transactions on Medical Imaging*, 27(4):495–508, 2008. [15](#), [55](#)
- [99] Manik Varma and Andrew Zisserman. Texture classification: Are filter banks necessary? In *International Conference on Computer vision and Pattern Recognition (CVPR)*, volume 2, pages II–691, 2003. [36](#), [46](#)
- [100] Konstantinos Veropoulos, Colin Campbell, Nello Cristianini, et al. Controlling the sensitivity of support vector machines. In *Proceedings of the international joint conference on AI*, pages 55–60, 1999. [29](#)
- [101] Luc Vincent and Pierre Soille. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 13(6):583–598, Jun 1991. [19](#)
- [102] Paul A. Viola and Michael J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004. [42](#), [66](#)
- [103] Shiv Vitaladevuni, Yuriy Mishchenko, Alex Genkin, Dmitri Chklovskii, and Kristen Harris. Mitochondria detection in electron microscopy images. In *Workshop on Microscopic Image Analysis with Applications in Biology*, 2008. [53](#)
- [104] Shuo Wang and Xin Yao. Diversity analysis on imbalanced data sets by using ensemble models. In *Computational Intelligence and Data Mining, 2009. CIDM'09. IEEE Symposium on*, pages 324–331. IEEE, 2009. [29](#)
- [105] Brian A. Wilt, Laurie D. Burns, Eric T. W. Ho, Kunal K. Ghosh, Eran A. Mukamel, and Mark J. Schnitzer. Advances in light microscopy for neuroscience. *Annual Review of Neuroscience*, 32(435), 2009. [53](#)
- [106] Hui Zou, Ji Zhu, Trevor Hastie, et al. New multicategory boosting algorithms based on multicategory fisher-consistent losses. *The Annals of Applied Statistics*, 2(4):1290–1306, 2008. [22](#), [23](#)