Project no. FP6-507752

MUSCLE

Network of Excellence
Multimedia Understanding through Semantics, Computation and LEarning

Progress on Semi-Supervised Machine Learning Techniques

Due date of deliverable: 31.09.2005
Actual submission date: 10.10.2005

Start date of project: 1 March 2004
Duration: 48 Months

Workpackage: 8
Deliverable: D8.1c (4/5)

Editors:
Pádraig Cunningham
Department of Computer Science
Trinity College Dublin
Dublin 2, Ireland

Revision 1.0

Project co-funded by the European Commission within the Sixth Framework Programme
(2002-2006)

<table>
<thead>
<tr>
<th>Dissemination Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>PU</td>
</tr>
<tr>
<td>PP</td>
</tr>
<tr>
<td>RE</td>
</tr>
<tr>
<td>CO</td>
</tr>
</tbody>
</table>

Keyword List:

©MUSCLE- Network of Excellence www.muscle-noe.org
### Contributing Authors

<table>
<thead>
<tr>
<th>Name</th>
<th>Institution</th>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nozha Boujemaa</td>
<td>INRIA-IMEDIA</td>
<td><a href="mailto:nozha.boujemaa@inria.fr">nozha.boujemaa@inria.fr</a></td>
</tr>
<tr>
<td>Michel Crucianu</td>
<td>INRIA-IMEDIA</td>
<td><a href="mailto:michel.crucianu@inria.fr">michel.crucianu@inria.fr</a></td>
</tr>
<tr>
<td>Pádraig Cunningham</td>
<td>TCD</td>
<td><a href="mailto:Padraig.Cunningham@cs.tcd.ie">Padraig.Cunningham@cs.tcd.ie</a></td>
</tr>
<tr>
<td>Nizar Grira</td>
<td>INRIA-IMEDIA</td>
<td><a href="mailto:nizar.grira@inria.fr">nizar.grira@inria.fr</a></td>
</tr>
<tr>
<td>Allan Hanbury</td>
<td>TU Wien</td>
<td><a href="mailto:hanbury@prip.tuwien.ac.at">hanbury@prip.tuwien.ac.at</a></td>
</tr>
<tr>
<td>Shaul Markovitch</td>
<td>Technion</td>
<td><a href="mailto:shaulm@cs.Technion.ac.il">shaulm@cs.Technion.ac.il</a></td>
</tr>
<tr>
<td>Branislav Mičušík</td>
<td>TU Wien</td>
<td><a href="mailto:micusik@prip.tuwien.ac.at">micusik@prip.tuwien.ac.at</a></td>
</tr>
</tbody>
</table>
## Contents

**WP7: State of the Art**  
1

Contents  
3

1 **Introduction**  
5

2 **One-Class Classification Problems**  
7  
2.1 One-class learning for image segmentation  
8  
2.1.1 Segmentation  
10  
2.1.2 Experiments  
12  
2.1.3 Conclusion  
14

3 **Active Learning**  
15  
3.1 Active Learning  
16  
3.2 The Source of the Examples  
17  
3.2.1 Example Construction  
17  
3.2.2 Selective Sampling  
17  
3.3 Example Evaluation and Selection  
18  
3.3.1 Uncertainty-based Sampling  
18  
3.3.2 Committee-based sampling  
19  
3.3.3 Clustering  
20  
3.3.4 Utility-Based Sampling  
20  
3.4 Conclusion  
21

4 **Semi-Supervised Clustering**  
23  
4.1 Semi-supervised Clustering of Images  
24  
4.2 Pairwise-Constrained Competitive Agglomeration  
25  
4.3 Active Selection of Pairwise Constraints  
26  
4.4 Experimental Evaluation  
28

5 **General Conclusions**  
31

Bibliography  
33
Chapter 1

Introduction

A fundamental distinction in machine learning research is that between unsupervised and supervised techniques. In supervised learning, the learning system is presented with labelled examples and induces a model from this data that will correctly label other unlabelled data. In unsupervised learning the data is unlabelled and all the learning system can do is organise or cluster the data in some way. In processing multimedia data, some important real world problems have emerged that do not fall into these two categories but have some characteristics of both: some examples are:

- One Class Classification Problems: abundant training data is available for one class of a binary classification problem but not the other (e.g. process monitoring, text classification).

- Active Learning: training data is unlabelled but a ‘supervisor’ is available to label examples at the request of the learning system. The system must select examples for labelling that make best use of the supervisor’s effort (e.g. labelling images for image retrieval).

- Semi-Supervised Clustering: it is often the case that background knowledge is available that can help in the clustering of data. Such extra knowledge can enable the discovery of small but significant bi-clusters in gene expression data. More generally, the use of background knowledge avoids situations where the clustering algorithm is obviously ‘getting it wrong’.

Since the Machine Learning (ML) research within the Muscle network is applications driven, there is considerable interest among the network members in these research issues. This report describes some initial research under each of these three titles and we can expect to see a good deal of activity in this area that lies between unsupervised and supervised ML techniques in the future.

In chapter 2 of this report we provide an overview of One-Class Classification problems and describe an application in image segmentation where foreground - background separation is treated in a ‘one-class’ framework. In chapter 3 an overview of Active Learning is presented; this overview is particularly focused on sample selection - the core research issue in Active Learning. Chapter 4 begins with an overview of Semi-Supervised Clustering and describes an some research from the network on the Semi-Supervised Clustering of images.
Chapter 2

One-Class Classification Problems

In traditional supervised classification problems, discriminating classifiers are trained using positive and negative examples. However, for a significant number of practical problems, counterexamples are either rare, entirely unavailable or statistically unrepresentative. Such problems include industrial process control, text classification and analysis of chemical spectra. One-sided classifiers have emerged as a technique for situations where labelled data exists for only one of the classes in a two-class problem. For instance, in industrial inspection tasks, abundant data may only exist describing the process operating correctly. It is difficult to put together training data describing the myriad of ways the system might operate incorrectly. A related problem is where negative examples exist, but their distribution cannot be characterised. For example, it is reasonable to provide characteristic examples of family pictures but impossible to provide examples of pictures that are ‘typical’ of non-family pictures. In this project we will implement a comprehensive toolkit for building one sided-classifiers.

These one-sided classification problems arise in a variety of situations:

- industrial inspection and process control
- text classification
- image labelling

A basic technique for developing one-class classifiers is to use Principal Component Analysis (PCA) or Independent Component Analysis (ICA). Details on PCA and ICA can be found in [65]. The basic principle is best explained with reference to Figure 2.1. With this approach, PCA is performed on the labelled training data. Essentially this transforms the data into the new space of principal components (PCs). These PCs will be ranked so that the first PC captures most of the variation in the data and so forth. It is normal to drop the lesser PCs as most of the information will be captured in the first two or three PCs. Figure 2.1 shows a cloud of examples described in a 3D space defined by the first 3 PCs. The different ellipsoids in the diagram define regions bounded by 1σ, 2σ and 3σ in the PCs. A classifier could operate by classifying any unlabeled example that falls within the 2σ ellipsoid as a positive example and everything outside as a negative example.

The problem with this is that PCA and ICA are unsupervised techniques for dimension reduction. So PCs are being discarded because the data varies little in those dimensions; it is
nevertheless possible that those dimensions are discriminating. Such a classifier will provide a baseline for comparison as this PCA based approach is commonly used in industry [71].

### 2.1 One-class learning for image segmentation

Image segmentation can be viewed as a partitioning of the image into regions having some similar properties, e.g. colour, texture, shape, etc., or as a partitioning of the image into semantically meaningful parts like people do. Moreover, measuring the goodness of segmentations in general is an unsolved problem and obtaining absolute ground truth is almost impossible since different people produce different manual segmentations of the same images [44].

There are many papers dealing with automatic segmentation. We have to mention the well known work of Shi & Malik [61] based on normalized cuts which segments the image into non-overlapping regions. They introduced a modification of graph cuts, namely normalized graph cuts, and provided an approximate closed-form solution. However, the boundaries of detected regions often do not follow the true boundaries of the objects. The work [64] is a follow-up to [61] where the segmentation is improved by doing it at various scales.

One possibility to partially avoid the ill-posed problem of image segmentation is to use additional constraints. Such constraints can be i) motion in the image caused either by camera motion or by motion of objects in the scene [56, 75, 4], or ii) specifying the foreground object properties [16, 52, 13].

Here we concentrate on an easier task than fully automatic segmentation. We constrain the segmentation by using a small user-provided template patch. We search for the segments of the image coherent in terms of colour with the provided template. The texture of an input image
Figure 2.2: Supervised segmentation. (a) Original image. Top image shows marked place from which the template was cut. (b) The enlarged template patch. (c) binary segmentation with masked original image.

is taken into account to correctly detect boundaries of textured regions. The proposed method gives the possibility to learn semi-automatically what the template patch should look like so as to be representative of an object leading to a correct one-class segmentation.

The proposed technique could be useful for segmentation and for detecting the objects in images with a characteristic a priori known property defined through a template patch. Fig. 2.2 shows how a tiger can be detected in the image using a small template patch from the same image. The same template patch can be used to detect the tiger in another image even though lighting conditions are slightly different, as shown in Fig. 2.2.

We follow the idea given in [16] of interactive segmentation where the user has to specify some pixels belonging to the foreground and to the background. Such labeled pixels give a strong constraint for further segmentation based on the min-cut/max-flow algorithm given in [17]. However, the method [16] was designed for greyscale images and thus most of the information is thrown away. We improved the method in [47] to cope with colour and texture images. However, both seeds for background and foreground objects were still needed. In this work we avoid the need of background seeds and only seeds for the foreground object need to be specified. This is therefore a one-class learning problem as we have a sample of what we are looking for (foreground), but the characteristics of the rest of the image (background) are unknown.

In [79] the spatial coherence of the pixels together with standard local measurements (intensity, colour) is handled. They propose an energy function that operates simultaneously in feature space and in image space. Some forms of such an energy function are studied in [35]. In
our work we follow a similar strategy. However, we define the neighborhood relation through brightness, colour and texture gradients introduced in [42, 45].

In the paper [52] a similar idea to ours has been treated. In principle, the result is the same, but the strategy to obtain it differs. The boundary of a textured foreground object is achieved by minimization (through the evolution of the region contour) of energies inside and outside the region. The Geodetic Active Region framework is used to propagate the region contour. However, the texture information for the foreground has to be specified by the user. In [51] the user interaction is omitted. At first the number of regions is estimated by fitting a mixture of Gaussians on the intensity histogram and is then used to drive the region evolution. However, such a technique cannot be used for textured images. One textured region can be composed of many colours and therefore Gaussian components say nothing about the number of dominant textures.

Our main contribution lies in incorporating the information included in the template patch into the graph representing the image, leading to a reasonable binary image segmentation. Our method does not need seeds for both foreground and the background as in [16, 47]. Only some representative template patch of the object being searched for is required, see Fig. 2.2. Moreover, texture information is taken into account.

This section is organised as follows. First, segmentation based on the graph cut algorithm is outlined. Second, non-parametric computation of probabilities of points being foreground/background through histograms and incorporating template patch information is described. Finally, the results and summary conclude the section.

2.1.1 Segmentation

We use the seed segmentation technique [47] based on the interactive graph cut method [16]. The core of the segmentation method is based on an efficient algorithm [17] for finding min-cut/max-flow in a graph. We very briefly outline the construction of the graph representing the image further used for finding min-cut of it. We describe in more detail the stage for building pixel penalties of being foreground or background.

Graph representing the image

The general framework for building the graph is depicted in Fig. 2.3 (left). The graph is shown for a 9 pixel image and an 8-point neighborhood $\mathcal{N}$. For general images, the graph has as many nodes as pixels plus two extra nodes labeled $F, B$. In addition, the pixel neighborhood is larger, e.g. we use a window of size $21 \times 21$ pixels.

For details on setting $W_{q,r}$, the reader is referred to [47, 48]. To fill this matrix the colour and texture gradient introduced in [42, 45] producing a combined boundary probability image is used. The reason is that our main emphasis is put on boundaries at the changes of different textured regions and not local changes inside one texture. However, there usually are large responses of edge detectors inside the texture. To detect boundaries in images correctly, the colour changes and texturedness of the regions have to be taken into account as in [42, 45].

Each node in the graph is connected to the two extra nodes $F, B$. This allows the incorporation of the information provided by the seed and a penalty for each pixel being foreground or
The foreground histogram is computed from all pixels in the template patch. Computation of the background histogram $b_{\{L,a,b\}}(i)$, resp. $b_{\{L,a,b\}}(i)$, represents the foreground, resp. the background histogram of each colour channel separately at the $i$th bin smoothed by a Gaussian kernel.

All histogram channels are smoothed using one-dimensional Gaussians, simply written as $\tilde{f}_i = \frac{1}{G} \sum_{j=1}^{N} f_j e^{-\frac{(j-i)^2}{2\sigma^2}}$, where $G$ is a normalization factor enforcing $\sum_{i=1}^{N} \tilde{f}_i = 1$. In our case, the number of histogram bins $N = 64$. We used $\sigma = 1$ since experiments showed that it is a reasonable value. $\lambda$ from the table in Fig. 2.3 was set to 1000.

The foreground histogram is computed from all pixels in the template patch. Computation of the background histogram $b_{\{L,a,b\}}$ is based on the following arguments.
Figure 2.4: Comparison of two histograms. The first one, shown in (a), is computed from all image pixels, in contrast to the second one, shown in (b), computed from template patch points. The template patch is cut out from the image at the position shown as a light square. The histograms are only schematic and are shown for one colour channel only to illustrate the idea described in the text.

We suggest to compute the background histogram from all image pixels since we know a priori neither the colours nor a template patch of the background. The basic idea behind this is the assumption that the histogram computed from all points includes information of all colours (the background and the foreground) in the image, see Fig. 2.4 for better understanding.

Assume that the texture is composed of two significant colours, like the template patch in Fig. 2.4(b) shows. Two significant colours create two peaks in the histograms. The same two peaks appear in both histograms. But, since \( \sum_{i=1}^{N} \tilde{b}_i = 1 \), the probability \( p(c_q|B) \) gives smaller values than \( p(c_q|F) \) for the colours present in the template. Thus, points more similar to the template are assigned in the graph more strongly to the foreground than to the background node. It causes the min-cut to go through the weaker edge with higher probability. However, the edges connecting the points in the neighbourhood play a role in the final cut as well. Therefore the words “higher probability”, because it cannot be said that the min-cut always cuts the weaker edge from two edges connecting the pixel with \( F, B \) nodes. An inherent assumption in this approach is that the surface area in the image of the region matching the texture patch is “small” compared to the area of the whole image. Experiments on the size of the region at which this assumption breaks down remain to be done.

2.1.2 Experiments

The segmentation method was implemented in MATLAB. Some of the most time consuming operations (such as creating the graph edge weights) were implemented in C and interfaced with MATLAB through mex-files. We used with advantage the sparse matrices directly offered by MATLAB. We used the online available C++ implementations of the min-cut algorithm \cite{17} and some MATLAB code for colour and texture gradient computation \cite{42}.

The most time consuming part of the segmentation process is creating the weight matrix \( \tilde{w} \). It takes 50 seconds on a 250 × 375 image running on a Pentium 4@2.8 GHz. The implementation of the texture gradient in C would dramatically speed up the computation time. Once the graph is built, finding the min-cut takes 2 – 10 seconds.

In all our experiments we used images from the Berkeley database \cite{32}. We marked a small “representative” part of the image and used it for further image segmentation of the image. See Fig. 2.5 for the results. From the results it can be seen that very good segmentation can be
Figure 2.5: Results (we recommend to see a colour version of this Figure). 1st column: enlarged image template patch. 2nd column: input image with marked area used as the template. 3rd column: binary segmentation. 4th column: segmentation with masked original image.
obtained even though only the colour histogram of the template patch is taken into account.

It is also possible to apply the template patch obtained from one image for segmenting another one. In the case depicted in Fig. 2.2 small tiger patch encoding the tiger’s colours obtained from one image is used for finding the tiger in another image. It can be seen that most of the tiger’s body was captured but also some pixels belonging to the background were segmented. Such “non-tiger” regions could be pruned using some further procedure, which is not discussed in this paper.

It opens new possibilities for the use of the method, e.g., for image retrieval applications. Since some representative image template is available, images from large databases coherent in colour and texture can be found. This application remains to be investigated.

2.1.3 Conclusion

We suggest a method for supervised texture segmentation in images using one-class learning. The method is based on finding the min-cut/max-flow in a graph representing the image to be segmented. We described how to handle the information present in a small representative template patch provided by the user. We proposed a new strategy to avoid the need for a background template or for a priori information of the background. The one-class learning problem is difficult in this case as no information about the background is known a priori. Experiments presented on some images from the Berkeley database show that the method gives reasonable results.

The method gives the possibility to detect objects in an image database coherent with a provided representative template patch. A question still to be considered is how the representative template patch for each class of interest should look. For example, the template patch should be invariant under illumination changes, perspective distortion, etc.
In many potential applications of machine learning it is difficult to get an adequate set of labelled examples to train a supervised machine learning system. This may be because:

- of the vast amount of training data available (e.g. astrophysical data, image retrieval or text mining),
- the labelling is expensive or time consuming (Bioinformatics or medical applications) or
- the labels are ephemeral as is the situation in assigning relevance in information retrieval.

In these circumstances an Active Learning (AL) methodology is appropriate, i.e. the learner is presented with a stream or a pool of unlabelled examples and may request a "supervisor" to assign labels to a subset of these. The idea of AL has been around since the early 90's (e.g. [59]); however, it remains an interesting research area today as there are many applications of AL that need to start learning with very small labeled data sets. The two motivating examples we will explore are the image indexing problem described in B.1.3 and the problem of bootstrapping a message classification system. Both of these scenarios suffer from the "small sample problem" that has been considered in relevance feedback in information retrieval systems.

Relevance feedback as studied in Information Retrieval [57] is closely related to Active Learning in the sense that both need to induce models from small samples. Relevance feedback (RF) has also been used in content based image retrieval (Rui et al. 1998). However, the important difference is that RF is searching for a target document or image whereas the objective with AL is to induce a classifier from a minimal amount of data. In addition RF is transductive rather than inductive in that the algorithm has access to the full population of examples. Also, in RF, examples are selected with two objectives in mind. It is expected that they will prove informative to the learner when labelled and it is hoped that they will satisfy the user's information need. Whereas in AL, examples are selected purely with the objective of maximising information for the learner.

This issue of example selection is at the heart of research on AL. The normal guiding principles are to select the unlabelled examples on which the learner is least certain or alternatively to select the examples that will prove most informative for the learner. These often boil down to the same thing. A classic strategy is that described by Tong and Koller [69] for Active Learning
with SVMs. They present a formal analysis based on version spaces that shows that a good policy for selecting samples for AL from a pool of data is to select examples that are closest to the separating hyperplane of the SVM. A better but more expensive policy is to select the example that would produce the smallest margin in the next version of SVM. Clearly these policies are less reliable when starting with a very small number of labelled examples as the version space is very large and the location of the hyperplane is unstable. It is also desirable to be able to select sets of examples together so it would be useful to introduce some notion of diversity to maximise the utility of these complete sets.

3.1 Active Learning

Learning is a process where an agent uses a set of experiences to improve its problem solving capabilities. In the passive learning model, the learner passively receives a stream of experiences and processes them. In the active learning model the learner has some control over its training experiences. Markovitch and Scott \[43\] define the information filtering framework which specifies five types of selection mechanisms that a learning system can employ to increase the utility of the learning process. Active learning can be viewed in this model as selective experience filter.

Why should a learner be active? There are two major reasons for employing experience selection mechanisms:

1. If some experiences have negative utility – the quality of learning would have been better without them – then it is obviously desirable to filter them out. For example, noisy examples are usually harmful.

2. Even when all the experiences are of positive utility, there is a need to be selective or to have control over the order in which they are processed. This is because the learning agent has bounded training resources which should be managed carefully.

In passive learning systems, these problems are either ignored, leading to inefficient learning, or are solved by relying on a human teacher to select informative training experiences \[76\]. It may sometimes be advantageous for a system to select its own training experiences even when an external teacher is available. Scott & Markovitch \[58\] argue that a learning system has an advantage over an external teacher in selecting informative examples because, unlike a teacher, it can directly access its own knowledge base. This is important because the utility of a training experience depends upon the current state of the knowledge base.

In this summary we concentrate on active learning for supervised concept induction. Most of the works in this area assume that there are high costs associated with tagging examples. Consider, for example, training a classifier for a character recognition problem. In this case, the character images are easily available, while their classification is a costly process requiring a human operator.

In the following subsections we differentiate between pool-based selective sampling and example construction approaches, define a formal framework for selective sampling algorithms and present various approaches for solving the problem.
3.2 The Source of the Examples

One characteristic in which active learning algorithms vary is the source of input examples. Some algorithms construct input examples, while others select from a pool or a stream of unlabeled examples.

3.2.1 Example Construction

Active learning by example construction was formalized and theoretically studied by Angluin [2]. Angluin developed a model which allows the learner to ask two types of queries: a membership query, where the learner selects an instance \( x \) and is told its membership in the target concept, \( f(x) \); and an equivalence query, where the learner presents a hypothesis \( h \), and either told that \( h \equiv f \), or is given a counterexample. Many polynomial time algorithms based on Anguin’s model have been presented for learning target classes such as Horn sentences [3] and multivariate polynomials [18]. In each case, a domain-specific query construction algorithm was presented.

In practical experiments with example construction, there are algorithms that try to optimize an objective function like information gain or generalization error, and derive expressions for optimal queries in certain domains [63, 66]. Other algorithms find classification borders and regions in the input space and construct queries near the borders [12, 30].

A major drawback of query construction was shown by Baum and Ladner [37]. When trying to apply a query construction method to the domain of images of handwritten characters, many of the images constructed by the algorithm did not contain any recognizable characters. This is a problem in many real-world domains.

3.2.2 Selective Sampling

Selective sampling avoids the problem described above by assuming that a pool of unlabeled examples is available. Most algorithms work iteratively by evaluating the unlabeled examples and select one for labeling. Alternatively it is assumed that the algorithms receives a stream of unlabeled examples, and for each example a decision is made (according to some evaluation criteria) whether to query for a label or not.

The first approach can be formalized as follows. Let \( X \) be an instance space, a set of objects described by a finite collection of attributes (features). Let \( f : X \rightarrow \{0, 1\} \) be a teacher (also called an expert) that labels instances as 0 or 1. A (supervised) learning algorithm takes a set of labeled examples, \( \{<x_1, f(x_1)>, \ldots, <x_n, f(x_n)>\} \), and returns a hypothesis \( h : X \rightarrow \{0, 1\} \). Let \( X \subseteq X \) be an unlabeled training set. Let \( D = \{<x_i, f(x_i)> : x_i \in X, i = 1, \ldots, n\} \) be the training data – a set of labeled examples from \( X \). A selective sampling algorithm \( S_L \), specified relative to a learning algorithm \( L \), receives \( X \) and \( D \) as input and returns an unlabeled element of \( X \).

The process of learning with selective sampling can be described as an iterative procedure where at each iteration the selective sampling procedure is called to obtain an unlabeled example and the teacher is called to label that example. The labeled example is added to the set of currently available labeled examples and the updated set is given to the learning procedure, which induces a new classifier. This sequence repeats until some stopping criterion is satisfied.
Active Learner($X, f()$):

1. $D \leftarrow \emptyset$.
2. $h \leftarrow L(\emptyset)$.
3. While stopping-criterion is not satisfied do:
   a) $x \leftarrow S_L(X, D)$ ; Apply $S_L$ and get the next example.
   b) $\omega \leftarrow f(x)$ ; Ask the teacher to label $x$.
   c) $D \leftarrow D \cup \{\langle x, \omega \rangle\}$ ; Update the labeled examples set.
   d) $h \leftarrow L(D)$ ; Update the classifier.
4. Return classifier $h$.

Figure 3.1: Active learning with selective sampling. Active Learner is defined by specifying stopping criterion, learning algorithm $L$ and selective sampling algorithm $S_L$. It then works with unlabeled data $X$ and teacher $f()$ as input.

This criterion may be a resource bound, $M$, on the number of examples that the teacher is willing to label, or a lower bound on the desired class accuracy. Adopting the first stopping criterion, the goal of the selective sampling algorithm is to produce a sequence of length $M$ which leads to a best classifier according to some given measure.

The pseudo code for an active learning system that uses selective sampling is shown in Figure 3.1.

3.3 Example Evaluation and Selection

Active learning algorithms that perform selective sampling differ from each other by the way they select the next example for labeling. In this section we present some common approaches to this problem.

3.3.1 Uncertainty-based Sampling

Most of the works in active learning select untagged examples that the learner is most uncertain about. Lewis and Gale [39] present a probabilistic classifier for text classification. They assume that a large pool of unclassified text documents is available. The probabilistic classifier assigns tag probabilities to text documents based on already labeled documents. At each iteration, several unlabeled examples that were assigned probabilities closest to 0.5 are selected for labeling. It is shown that the method significantly reduces that amount of labeled data needed to achieve a certain level of accuracy, compared to random sampling. Later, Lewis and Catlett [38] used an efficient probabilistic classifier to select examples for training another (C4.5) classifier. Again, examples with probabilities closest to 0.5 were chosen.

Special uncertainty sampling methods were developed for specific classifiers. Park [53] uses sampling-at-the-boundary method to choose support vectors for SVM classifier. Orthogonal
support vectors are chosen from the boundary hyperplane and queried one at a time. Tong and Koller [70] also perform pool-based active learning of SVMs. Each new query is supposed to split the current version space into two parts as equal as possible. The authors suggest three methods of choosing the next example based on this principle. Hasenjager & Ritter[30] and Lindenbaum et. al [41] present selective sampling algorithms for nearest neighbor classifiers.

3.3.2 Committee-based sampling

Query By Committee (QBC) is a general uncertainty-based method where a committee of hypotheses consistent with the labeled data is specified and an unlabeled example on which the committee members most disagree is chosen. Seung, Oper and Sompolinsky [60] use Gibbs algorithm to choose random hypotheses from the version space for the committee, and choose an example on which the disagreement between the committee members is the biggest. Application of the method is shown on the high-low game and perceptron learning. Later, Freund et. al. [22] presented a more complete and general analysis of query by committee, and show that the method guarantees rapid decrease in prediction error for the perceptron algorithm.

Cohn et. al. [19] present the concept of uncertainty regions - the regions on which hypotheses that are consistent with the labeled data might disagree. Although it is difficult to exactly calculate there regions, one can use approximations (larger or smaller regions). The authors show an application of the method to neural networks - two networks are trained on the labeled example set, one is the most general network, and the other is the most specific network. When examining an unlabeled example, if the two networks disagree on the example, the true label is queried.

Krogh and Vedelsby [36] show the connection between the generalization error of a committee of neural networks and its ambiguity. The larger is the ambiguity, the smaller is the error. The authors build an ambiguous network ensemble by cross validation and calculate optimal weights for ensemble members. The ensemble can be used for active learning purposes - the example chosen for labeling is the one that leads to the largest ambiguity. Rayandhuri et. al. [55] use the same principle and analyze the results with an emphasize on minimizing data gathering. Hasenjager and Ritter [29] compare the theoretical optimum (maximization of information gain) of the high-low game with the performance of QBC approach. It is shown that QBC results rapidly approach the optimal theoretical results.

In some cases, especially in the presence of noise, it is impossible to find hypotheses consistent with the labeled data. Abe and Mamitsuka [1] suggest to use QBC with combination of boosting or bagging. Liere and Tadepalli [40] use a small committee of 7 Winnow classifiers for text categorization. Engelson and Dagan [5] overcome the problem of finding consistent hypotheses by generating a set of classifiers according to the posterior distribution of the parameters. The authors show how to do that for binomial and multinomial parameters. This method is only applicable when it is possible to estimate a posterior distribution over the model space given the training data. Muslela et. al. [49] present an active learning scheme for problems with several redundant views – each committee member is based on one of the problem views.

Park et. al. [54] combine active learning with using unlabeled examples for induction. They train a committee on an initial training set. At each iteration, the algorithm predicts the label of an unlabeled example. If prediction disagreement among committee members is below a
threshold, the label is considered to be the true label and committee members weights are updated according to their prediction. If disagreement is high, the algorithm queries the oracle (expert) for the true label. Baram et. al. [7] use a committee of known-to-be-good active learning algorithms. The committee is then used to select the next query. At each stage, the best performing algorithm in the committee is traced, and committee members weights are updated accordingly.

Melville and Mooney [46] use artificially created examples to construct highly diverse committees. Artificial examples are created based on the training set feature distribution, and labeled with the least probable label. A new classifier is built using the new training set, and if it does not increase the committee error, it is added to the committee.

### 3.3.3 Clustering

Recently, several works suggest performing pre-clustering of the instance pool before selecting an example for labeling. The motivation is that grouping similar examples may help to decide which example will be useful when labeled. Engelbrecht et. al. [21] cluster the unlabeled data and then perform selective sampling in each of the clusters. Nguyen et. al. [50] pre-cluster the unlabeled data and give priority to examples that are close to classification boundary and are representatives of dense clusters.

Soderland [62] incorporates active learning in an information extraction rules learning system. The system maintains a group of learned rules. In order to choose the next example for label, all unlabeled examples are divided to three categories - examples covered by rules, near misses, and examples not covered by any rule. Examples for labeling are sampled randomly from these groups (proportions are selected by the user).

### 3.3.4 Utility-Based Sampling

Fujii et. al. [24] have observed that considering only example uncertainty is not sufficient when selecting the next example for label, and that the impact of example labeling on other unlabeled examples should also be taken into account. Therefore, example utility is measured by the sum of interpretation certainty differences for all unlabeled examples, averaged with the examined example interpretation probabilities. The method was tested on the word sense disambiguation domain, and showed to perform better than other sampling methods.

Lindenbaum et. al. [41] present a very general utility-based sampling method. The sampling process is viewed as a game where the learner actions are the possible instances and the teacher reactions are the different labels. The sampling algorithm expands a “game” tree of depth $k$, where $k$ depends on the resources available. This tree contains all the possible sampling sequences of length $k$. The utility of each leaf is computed by evaluating the classifier resulting from performing induction on the sequence leading to that leaf. The arcs associated with the teacher’s reactions are assigned probabilities using a random field model. The example selected is the one with the highest expected utility.
3.4 Conclusion

Active learning is a powerful tool. It has been shown to be efficient in reducing the number of labeled examples needed to perform learning, thus reducing the cost of creating training sets. It has been successfully applied to many problem domains, and is potentially applicable to any task involving learning. Active learning was applied to numerous domains such as function approximation [36], speech recognition [72], recommender system [67], text classification and categorization [39, 70, 40, 68, 80], part of speech tagging [5], image classification [49] and word sense disambiguation [24, 54].
Chapter 4

Semi-Supervised Clustering

It would be generally accepted that conventional approaches to clustering have the draw-back that they do not bring background knowledge to bear on the clustering process [74] [11] [77]. This is particularly an issue in the analysis of gene-array data where clustering is a key data-analysis technique [78] [6]. This issue is increasingly being recognised and there is significant research activity that attempts to make progress in this area. Bolshakova et al [14] have shown how background information from Gene Ontologies can be used for cluster validity assessment. However, if extra information is available, it should be used directly in the determination of the clusters rather than after the fact in validation. The idea that background knowledge should be brought to bear in clustering is a general principle and there is a variety of ways in which this might be achieved in practice. Yip et al. [78] suggest there are three aspects to this:

- what extra knowledge is being input to the clustering process,
  - cluster labels may be available for some objects,
  - more commonly, information will be available on whether objects must-link or cannot-link.

- when the knowledge is input
  - before clustering to guide the clustering process,
  - after clustering to evaluate the clusters or guide the next round of clustering,
  - users may supply knowledge as the clustering proceeds.

- how the knowledge affects the clustering process
  - through the formation of seed clusters,
  - constraining some objects to be put in the same cluster or different clusters,
  - modifying the assessment of similarity/difference.

It is clear that a key determiner of the best approach to semi-supervised clustering is the nature of the external information that is being brought to bear.
4.1 Semi-supervised Clustering of Images

To let users easily apprehend the contents of image databases and to make their access to the images more effective, a relevant organisation of the contents must be achieved. The results of a fully automatic categorization (unsupervised clustering or cluster analysis, see the surveys in [34], [33]), relying exclusively on a measure of similarity between data items, are rarely satisfactory. Some supervision information provided by the user is then needed for obtaining results that are closer to user’s expectations. Supervised classification and semi-supervised clustering are both candidates for such a categorization approach. When the user is able and willing to provide class labels for a significant sample of images in the database, supervised classification is the method of choice. In practice, this will be the case for specific classification/recognition tasks, but not so often for database categorization tasks.

When the goal is general image database categorization, not only the user does not have labels for images, but he doesn’t even know a priori what most of the classes are and how many classes should be found. Instead, he expects the system to “discover” these classes for him. To improve results with respect to what an unsupervised algorithm would produce, the user may accept to provide some supervision if this information is of a very simple nature and in a rather limited amount. A semi-supervised clustering approach should then be employed. Semi-supervised clustering takes into account both the similarity between data items and information regarding either the membership of some data items to specific clusters or, more often, pairwise constraints (must-link, cannot-link) between data items. These two sources of information are combined ([26], [10]) either by modifying the search for appropriate clusters (search-based methods) or by adapting the similarity measure (similarity-adapting methods). Search-based methods consider that the similarities between data items provide relatively reliable information regarding the target categorization, but the algorithm needs some help in order to find the most relevant clusters. Similarity-adapting methods assume that the initial similarity measure has to be significantly modified (at a local or a more global scale) by the supervision in order to reflect correctly the target categorization. While similarity-adapting methods appear to apply to a wider range of situations, they need either significantly more supervision (which can be an unacceptable burden for the user) or specific strong assumptions regarding the target similarity measure (which can be a strong limitation in their domain of application).

We assume that users can easily evaluate whether two images should be in the same category or rather in different categories, so they can easily define must-link or cannot-link constraints between pairs of images. Following previous work by Demiriz et al. [20], Wagstaff et al. [73] or Basu et al. [8], in [28] we introduced Pairwise-Constrained Competitive Agglomeration (PCCA), a fuzzy semi-supervised clustering algorithm that exploits the simple information provided by pairwise constraints. PCCA is reminded in section 4.2. In the original version of PCCA [28] we did not make further assumptions regarding the data, so the pairs of items for which the user is required to define constraints are randomly selected. But in many cases, such assumptions regarding the data are available. We argue in [27] that quite general assumptions let us perform a more adequate, active selection of the pairs of items and thus significantly reduce the number of constraints required for achieving a desired level of performance. We present in section 4.3 our criterion for the active selection of constraints. Experimental results obtained with this criterion on a real-world image categorization problem are given in section 4.4.
4.2 Pairwise-Constrained Competitive Agglomeration

PCCA belongs to the family of search-based semi-supervised clustering methods. It is based on the Competitive Agglomeration (CA) algorithm [23], a fuzzy partitional algorithm that does not require the user to specify the number of clusters to be found. Let $x_i$, $i \in \{1, \ldots, N\}$ be a set of $N$ vectors representing the data items to be clustered, $V$ the matrix having as columns the prototypes $\mu_k$, $k \in \{1, \ldots, C\}$ of $C$ clusters ($C \ll N$) and $U$ the matrix of the membership degrees, such as $u_{ik}$ is the membership of $x_i$ to the cluster $k$. Let $d(x_i, \mu_k)$ be the distance between the vector $x_i$ and the cluster prototype $\mu_k$. For PCCA, the objective function to be minimized must combine the feature-based similarity between data items and knowledge of the pairwise constraints. Let $\mathcal{M}$ be the set of available must-link constraints, i.e. $(x_i, x_j) \in \mathcal{M}$ implies $x_i$ and $x_j$ should be assigned to the same cluster, and $\mathcal{C}$ the set of cannot-link constraints, i.e. $(x_i, x_j) \in \mathcal{C}$ implies $x_i$ and $x_j$ should be assigned to different clusters. With these notations, we can write the objective function PCCA must minimize [28]:

$$
J(V, U) = \sum_{k=1}^{C} \sum_{i=1}^{N} (u_{ik})^2 d^2(x_i, \mu_k) - \beta \sum_{k=1}^{C} \left[ \sum_{i=1}^{N} (u_{ik}) \right]^2
$$

$$
\quad + \alpha \left( \sum_{(x_i, x_j) \in \mathcal{M}} \sum_{k=1, l=1, l \neq k}^{C} u_{ik} u_{jl} + \sum_{(x_i, x_j) \in \mathcal{C}} \sum_{k=1}^{C} u_{ik} u_{jk} \right)
$$

under the constraint $\sum_{k=1}^{C} u_{ik} = 1$, for $i \in \{1, \ldots, N\}$. The prototypes of the clusters, for $k \in \{1, \ldots, C\}$, are given by

$$
\mu_k = \frac{\sum_{i=1}^{N} (u_{ik})^2 x_i}{\sum_{i=1}^{N} (u_{ik})^2}
$$

and the fuzzy cardinalities of the clusters are $N_k = \sum_{i=1}^{N} u_{ik}$.

The first term in (4.1) is the sum of the squared distances to the prototypes weighted by the memberships and comes from the FCM objective function; it reinforces the compactness of the clusters. The second term is composed of the cost of violating the pairwise must-link and cannot-link constraints. The penalty corresponding to the presence of two points in different clusters (for a must-link constraint) or in a same cluster (for a cannot-link constraint) is weighted by their membership values. The term taking the constraints into account is weighted by $\alpha$, a constant factor that specifies the relative importance of the supervision. The last term in (4.1) is the sum of the squares of the cardinalities of the clusters (coming from the CA objective function) and controls the competition between clusters.

When all these terms are combined and $\beta$ is well chosen, the final partition will minimize the sum of intra-cluster distances, while partitioning the data set into the smallest number of clusters such that the specified constraints are respected as well as possible. Note that when the memberships are crisp and the number of clusters is pre-defined, this cost function reduces to the one used by the PCKmeans algorithm in [8].

It can be shown (see [28]) that the equation for updating memberships is

$$
u_{rs} = u_{rs}^{\text{FCM}} + u_{rs}^{\text{Constraints}} + u_{rs}^{\text{Bias}}
$$

The first term in (4.3), $u_{rs}^{\text{FCM}}$, comes from FCM and only focusses on the distances between data items and prototypes. The second term, $u_{rs}^{\text{Constraints}}$, takes into account the supervision:
memberships are reinforced or deprecated according to the pairwise constraints given by the user. The third term, \( u_{i}^{Bias} \), leads to a reduction of the cardinality of spurious clusters, which are discarded when their cardinality drops below a threshold.

The \( \beta \) factor controls the competition between clusters and is defined at iteration \( t \) by:

\[
\beta(t) = \eta_0 \exp\left(\frac{-|t - t_0|/\tau}{\sum_{j=1}^{C} \left( \sum_{i=1}^{N} u_{ij}^2 \right)} \right) \sum_{j=1}^{C} \left( \sum_{i=1}^{N} u_{ij}^2 \right) \left( \sum_{j=1}^{N} d^2(x_i, \mu_j) \right)
\]

The exponential component of \( \beta \) makes the last term in (4.1) dominate during the first iterations of the algorithm, in order to reduce the number of clusters by removing spurious ones. Later, the first three terms will dominate, to seek the best partition of the data. The resulting PCCA algorithm is given in [28]. In the original version of PCCA, the pairs of items for which the user is required to define constraints are randomly selected, prior to running the clustering process. As distance \( d(x_i, \mu_j) \) between a data item \( x_i \) and a cluster prototype \( \mu_j \) one can use either the ordinary Euclidean distance when the clusters are assumed to be spherical or the Mahalanobis distance when they are assumed to be elliptical.

4.3 Active Selection of Pairwise Constraints

To make this semi-supervised clustering approach attractive for the user, it is important to minimize the number of constraints he has to provide for reaching some given level of quality. This can be done by asking the user to define must-link or cannot-link constraints for the pairs of data items that are expected to have the strongest corrective effect on the clustering algorithm (i.e. that are maximally informative).

But does the identity of the constraints one provides have a significant impact on performance or all constraints are relatively alike and only the number of constraints matters? In a series of repeated experiments with PCCA using random constraints, we found a significant variance in the quality of the final clustering results. So the selection of constraints can have a strong impact and we must find appropriate selection criteria. Such criteria may depend on further assumptions regarding the data; for the criteria to be relatively general, the assumptions they rely on should not be too restrictive.

In previous work on this issue, Basu et al. [9] developed a scheme for selecting pairwise constraints before running their semi-supervised clustering algorithm (PCKmeans). They defined a farthest-first traversal scheme of the set of data items, with the goal of finding \( k \) items that are far from each other to serve as support for the constraints. This issue was also explored in unsupervised learning for cases where prototypes cannot be defined. In such cases, clustering can only rely on the evaluation of pairwise similarities between data items, implying a high computational cost. In [31], the authors consider subsampling as a solution for reducing cost and perform an active selection of new data items by minimizing the estimated risk of making wrong predictions regarding the true clustering from the already seen data. This active selection method can also be seen as maximizing the expected value of the information provided by the new data items. The authors find that their active unsupervised clustering algorithm spends more samples of data to disambiguate clusters that are close to each other and less samples for items belonging to well-separated clusters.
As for other search-based semi-supervised clustering methods, when using PCCA we consider that the similarities between data items provide relatively reliable information regarding the target categorization and the constraints only help in order to find the most relevant clusters. There is then little uncertainty in identifying well-separated compact clusters. To be maximally informative, supervision effort (i.e. constraints) should rather be spent for defining those clusters that are neither compact, nor well-separated from their neighbors. One can note that this is consistent with the findings in [31] regarding unsupervised clustering. To find the data items that provide the most informative pairwise constraints, we shall then focus on the least well-defined clusters and, more specifically, on the frontier with their neighbors. To identify the least well-defined cluster at some iteration \( t \), we use the fuzzy hypervolume \( FHV = |C_k| \), with \(|C_k|\) the determinant of the covariance matrix \( C_k \) of cluster \( k \). The FHV was introduced by Gath and Geva [25] as an evaluation of the compactness of a fuzzy cluster; the smaller the spatial volume of the cluster and the more concentrated the data items are near its center, the lower the FHV of the cluster. We consider the least well-defined cluster at iteration \( t \) to be the one with the largest FHV at that iteration. Note that we don’t need any extra computation because we already used the FHV to find the Mahalanobis distance.

Once the least well-defined cluster at iteration \( t \) is found, we need to identify the data items near its boundary. Note that in the fuzzy setting, one can consider that a data item represented by the vector \( x_r \) is assigned to cluster \( s \) if \( u_{rs} \) is the highest among its membership degrees. The data items at the boundary are those having the lowest membership values to this cluster among all the items assigned to it. After obtaining a set of items that lie on the frontier of the cluster, we find for each item the closest cluster, corresponding to its second highest membership value. The user is then asked whether one of these items should (or not) be in the same cluster as the closest item from the nearest cluster. It is easy to see that the time complexity of this method is high. We suggest below an approximation of this method, with a much lower cost.

After finding the least well-defined cluster with the FHV criterion, we consider a virtual boundary that is only defined by a membership threshold and will usually be larger than the true one (this is why we call it “extended” boundary). The items whose membership values are closest to this threshold are considered to be on the boundary and constraints are generated directly between these items. We expect these constraints to be relatively equivalent (and not too suboptimal when compared) to the constraints that would have been obtained by the more complex method described in the previous paragraphs.

This approximate selection method has two parameters: the number of constraints at every iteration and the membership threshold for defining the boundary. The first parameter concerns the selection in general, not the approximate method specifically. In all the experiments presented next, we generate 3 constraints at every iteration of the clustering algorithm. For the comparative evaluations, we plot the ratio of well-categorized items against the number of pairwise constraints. With this selection procedure, when the maximal number of constraints is reached, clustering continues until convergence without generating new constraints. The second parameter is specific to the approximation of the boundary and we use a fixed value of 0.3, meaning that we consider a rather large approximation. These fixed values for the two parameters are necessarily suboptimal, but a significant increase in performance (or reduction in the number of required constraints for a given performance) is nevertheless obtained.
4.4 Experimental Evaluation

We evaluated the effect our criterion for the active selection of constraints has on the PCCA algorithm and we compared it to the CA algorithm [23] (unsupervised clustering) and to PCK-means [8] (semi-supervised clustering).

The comparison shown here is performed on a ground truth database composed of images of different phenotypes of *Arabidopsis thaliana*, corresponding to slightly different genotypes. This scientific image database is issued from studies of gene expression. There are 8 categories, defined by visual criteria: textured plants, plants with long stems and round leaves, plants with long stems and fine leaves, plants with dense, round leaves, plants with desiccated or yellow leaves, plants with large green leaves, plants with reddish leaves, plants with partially white leaves. There are a total of 187 plant images, but different categories contain very different numbers of instances. The intra-class diversity is high for most classes. A sample of the images is shown in [27]. The global image features we used are the Laplacian weighted histogram, the probability weighted histogram, the Hough histogram, the Fourier histogram and a classical color histogram obtained in HSV color space (described in [15]). Linear PCA was used for a five-fold reduction of the dimension of the feature vectors. In all the experiments reported here we employed the Mahalanobis distance rather than the classical Euclidean distance.

Figure 4.1 presents the dependence between the percentage of well-categorized data points and the number of pairwise constraints considered. We provide as a reference the graphs for the CA algorithm and for K-means, both ignoring the constraints (unsupervised learning). The correct number of classes was directly provided to K-means and PCK-means. CA, PCCA and Active-PCCA were initialized with a significantly larger number of classes and found the appropriate number by themselves.

For the fuzzy algorithms (CA, PCCA and Active-PCCA) every data item is assigned to the cluster to which its membership value is the highest. For every number of constraints, 500 experiments were performed with different random selections of the constraints in order to produce the error bars for PCK-means and for the original PCCA.

These experimental results clearly show that the user can significantly improve the quality of the categories obtained by providing a simple form of supervision, the pairwise constraints. With a similar number of constraints, PCCA performs significantly better than PCK-means by making a better use of the available constraints. The fuzzy clustering process directly takes into account the the pairwise constraints thanks to the signed constraint terms in the equation for updating the memberships. The active selection of constraints (Active-PCCA) further reduces the number of constraints required for reaching such an improvement. The number of constraints becomes very low with respect to the number of items in the dataset.

By including information provided by the user, semi-supervised image database categorization can produce results that come much closer to user’s expectations. But the user will not accept such an approach unless the information he must provide is very simple in nature and in a small amount. The experiments we performed show that the active selection of constraints combined with PCCA allows the number of constraints required to remain sufficiently low for this approach to be a valuable alternative in the categorization of image databases. The approximations we made allowed us to maintain a low computational complexity for the resulting algorithm, making it suitable for real-world clustering applications.
Figure 4.1: Results obtained by the different clustering algorithms on the Arabidopsis database

This work was supported by the EU Network of Excellence (http://www.muscle-noe.org/). The authors wish also to thank NASC (European Arabidopsis Stock Centre, UK, http://arabidopsis.info/) for allowing them to use the Arabidopsis images and Ian Small from INRA (Institut National de la Recherche Agronomique, France) for providing this image database.
Chapter 5

General Conclusions

A special characteristic of the Machine Learning research in Muscle is that it is applications driven. It is driven by the characteristics of scenarios in which ML techniques might be used in processing multimedia data. The role for ML in these situations often does not fit well with standard ML techniques that are divided into those that deal with labeled data and those that do not. In this report we have identified three distinct variants of scenarios that do not fit the classic supervised/unsupervised organisation:

**One Sided Classifiers** Sometimes it is in the nature of the problem that labeled examples are available for one class only. This situation is described in detail in Chapter 2 and an approach for tackling image foreground-background separation as a one-class classification problem is described in Section 2.1.

**Active Learning** It is often the case that producing a corpus of labeled training examples is an expensive process. This is the situation for instance when supervised learning techniques are used for annotating images in a collection. Active Learning refers to a general set of techniques for building classifiers while minimising the number of labeled examples required. The core research question in Active Learning is the problem of selecting the most useful examples to present to the user for labeling. Promising approaches for this are presented in Chapter 3.

**Semi-Supervised Clustering** Confidence is a clustering system will be damaged if it returns clusters that disagree with background knowledge a user might have about the objects under consideration. Thus it is very desirable to be able to present this background knowledge as an input to the clustering process. A natural form for this background knowledge to take is as *must-link* and *cannot-link* constraints that indicate objects that should and should not be grouped together. PCCA as described in section 4.2 is a framework that allows background knowledge expressed as pairwise constraints to influence the clustering process. This idea is further enhanced in section 4.3 where a process for identifying useful pairwise constraints is introduced. This *active* selection of constraints is in the spirit of Active Learning as described in Chapter 3.

This research on ML techniques that fall somewhere between the traditional unsupervised and supervised alternatives is a key area of work within Muscle. This is because many of
the techniques under consideration are specialised for processing multimedia data. For this reason there will be further activity in the area within the network and we should expect further innovation in this area.
Bibliography


[37] K.J. Lang and E.B. Baum. Query learning can work poorly when a human oracle is used. In International Joint Conference on Neural Networks, Septebmer 1992.


