Segmentation with Incremental Classifiers

Guillaume Bernard[1,2], Michel Verleysen[2], and John A. Lee[1,2,⋆]

[1] Molecular Imaging, Radiotherapy, and Oncology – IREC
[2] Machine Learning Group – ICTEAM
Université catholique de Louvain, Belgium

Abstract. Radiotherapy treatment planning requires physicians to delineate the target volumes and organs at risk on 3D images of the patient. This segmentation task consumes a lot of time and can be partly automated with atlases (reference images segmented by experts). To segment any new image, the atlas is non-rigidly registered and the organ contours are then transferred. In practice, this approach suffers from the current limitations of non-rigid registration. We propose an alternative approach to extract and encode the physician's expertise. It relies on a specific classification method that incrementally extracts information from groups of pixels in the images. The incremental nature of the process allows us to extract features that depend on partial classification results but also convey richer information. This paper is a first investigation of such an incremental scheme, illustrated with experiments on artificial images.

1 Introduction

Cancer treatment with radiation beams amounts to a ballistic problem where the dose to the tumor must be maximized while the dose at surrounding healthy tissues must be minimized to avoid secondary effects. In order to achieve the best tradeoff, 3D images of the patients must be segmented to identify the tumor and the organs at risk. The physicians use an electronic pen or a mouse to delineate these volumes on each slice. Although it consumes a lot of time, delineation usually remains manual because it involves complex expertise. This explains why usual image segmentation methods such as histogram thresholding [11], pixel or patch clustering [12], gradient peak detection with active contours [9], or watersheds [2,5] cannot solve the problem. Many of these methods are unsupervised, even though some of them can take into account some a priori information, such as the expected region shape, size, and edge smoothness. On the other hand, supervised segmentation remains difficult to apply, mainly because the encoding of expertise and a priori information is far from being trivial. The most successful approach is the use of atlases, which are (banks of) images that are segmented beforehand by experts. Atlases can be deformed to match any new image with a non-rigid registration algorithm [3]. Once the two images are aligned, the contours or regions can be propagated from the atlas to the new image. This approach suffers from the shortcomings of the registration algorithms

⋆ J.A.L. is a Research Associate with the Belgian F.R.S.-FNRS.

A. Petrosino (Ed.): ICIAP 2013, Part II, LNCS 8157, pp. 81–90, 2013.
© Springer-Verlag Berlin Heidelberg 2013

it relies on. Many of these algorithms regularize the deformation vector field in a simplistic or unrealistic way. This leads to segmentation results that are globally correct but often inaccurate near the region boundaries; the required corrections annihilate the expected gain of time.

From a theoretical point of view, the segmentation of several objects in an image amounts to a supervised multiclass classification problem. In practice, however, this alternative approach faces several obstacles, the most prominent being that usual classification algorithms can only deal with features that are class-independent and thus *intrinsic* to the image, such as pixel coordinates, pixel luminance, or patch textures. These features convey limited information about the objects depicted in the images. On the other hand, *extrinsic* features that describe the relationships between two or more classes in the image have a richer content but they are more difficult to take into account. For instance, let us consider a feature such as the spatial distance to the region of class Y in the image. When training a classifier, this feature can be trivially computed since the labels of all regions are known in a pre-segmented image. In contrast, in a test image, measuring this distance requires at least some pixels to be already given label Y. A pragmatic solution to take benefit of extrinsic features consists in stacking at least two classifiers. The first one involves only intrinsic features. The resulting partial classification can then serve to compute a first batch of extrinsic features, which are fed into a second classifier, and so on. This incremental process has been investigated in [7], for example.

This paper suggests a generic approach, where the images are first over-segmented with a watershed transform. Information extracted from the watersheds are used for the multiclass problem, which is first divided into several binary classification problems (one class versus all others). Binary classifiers (k nearest neighbors [6], support vector machine [8] and random forest [4]) tackle these problems repeatedly in an iterative way. Two methods are proposed to select the order in which the binary classifiers should be run. At the end of this incremental process, a multiclass classifier is used with all computed features, to improve the final results. The efficacy of the approach is demonstrated in a few segmentation tasks involving artificial images.

This paper is organized as follows. Section 2 briefly describes the method used for the unsupervised over-segmentation of the images. Section 3 introduces the notations for intrinsic, extrinsic, and known features; it also details the two proposed methods of feature ranking. Section 4 describes the incremental procedure for feature computation and partial classification, as well as the final multiclass classification. Section 5 reports and discusses the experimental results. Finally, Section 6 draws the conclusions.

2 Unsupervised Over-segmentation

In order to obtain a first, unsupervised segmentation of the images, a watershed transform is used [2,5,1]. The principle is to consider the gradient magnitude image as a topographic relief where a flooding is simulated. The dam lines

separating the catchment basins yield an over-segmentation of the image. This preprocessing step limits the computational complexity by working with consistent groups of similar pixels, called *super-pixels*, rather than with pixels themselves. To control the over-segmentation granularity, we use a reformulation of the watershed transform as a graph-cut problem, such as described in [5] and [10]. This watershed-cut method works with both 2D and 3D images and it includes a graph filtering step that affects the number of super-pixels.

3 Intrinsic, Extrinsic, and Known Features

Let $\{\mathbf{X}, \mathbf{y}\}$ denote a data set where $\mathbf{X} = [x_{ij}]_{1 \leq i \leq D, 1 \leq j \leq N}$ contains the features and $\mathbf{y} = [y_j]_{1 \leq j \leq N}$ gives the corresponding labels. Label y_j takes its value in $\{Y_1, Y_2, \ldots, Y_C\}$, where C is the number of classes.

In the training phase, the rows of data set $\mathbf{X}$ can be distinguished between intrinsic and extrinsic features. Let $\mathcal{F}_{\text{in}}$, $\mathcal{F}_{\text{ex}}$ denote the non-intersecting sets of indices corresponding to intrinsic and extrinsic features. As extrinsic features represent relationships between objects of different classes, we assume that N is a multiple of C and that the data set consists of N/C groups of objects where all classes are instantiated at least once. Within the framework of image segmentation, this means that each image contains at least an object of each kind. This assumption avoids undetermined or ambiguous relationships.

In the test phase, the unlabeled data set $\mathbf{X}'$ contains missing values for all extrinsic features. During the incremental classification process, blanks are filled in as soon as class labels are attributed. Let $\mathcal{F}_{\text{kn}}^{(t)}$ denote the set of indices corresponding to known features at iteration t of the incremental procedure. We have $\mathcal{F}_{\text{in}} = \mathcal{F}_{\text{kn}}^{(1)} \subseteq \mathcal{F}_{\text{kn}}^{(t)} \subseteq \mathcal{F}_{\text{kn}}^{(t+1)}$.

As the evaluation of yet unknown extrinsic features requires a certain class label to be attributed with reasonable certainty, it is more natural to use binary classifiers that are specialized for the considered class. Therefore, an execution sequence of the binary classifiers has to be determined. For this purpose, the already known features must be ranked according to their usefulness for binary classification. This paper proposes two methods to rank the features.

3.1 Feature Ranking by Nearest Neighbors

As a first method, we suggest a ranking that assesses the overlap of classes for a given feature. Let $\mathcal{N}_j^K(\mathbf{X})$ denote the set of indices corresponding to the K nearest neighbors of the j^{th} super-pixel $\mathbf{x}_{\bullet j}$ of data set $\mathbf{X}$. Let $\mathcal{C}_k(\mathbf{y}) = \{p \text{ s.t. } y_p = Y_k\}$ be the set of indices associated with class Y_k. The usefulness of a certain feature to classify data inside or outside class Y_k can be measured as

$$s_{ik} = \frac{1}{K|\mathcal{C}_k(\mathbf{y})|} \sum_{p \in \mathcal{C}_k(\mathbf{y})} \left| \{q \text{ s.t. } q \in \mathcal{N}_p^K(\mathbf{e}_i^T \mathbf{X}) \text{ and } y_q = Y_k\} \right| ,$$

where $|A|$ denotes the cardinality of set A and $\mathbf{e}_i$ is a vector of zeros everywhere except the ith element equal to 1. The value of s_{ik} can range from 0 to 1.

The latter value indicates that class Y_k does not overlap with other classes along the axis of the ith feature. Each row of matrix $\mathbf{S} = [s_{ik}]$ can be computed quite efficiently by sorting vector $\mathbf{e}_i^T\mathbf{X}$ and sliding a $(2K+1)$-wide window. This leads to a computational complexity of $\mathcal{O}(N\ln(N) + NK\ln(K))$ for each feature. The following steps need to be realized to obtain the binary classification order:

1. $\mathcal{F}_{\text{kn}} = \mathcal{F}_{\text{in}}$; O is an empty 'first in first out' list (FIFO).
2. Compute $\ell = \max_k s_{ik}$ with $i \in \mathcal{F}_{\text{kn}}$; set $s_{\bullet\ell} = 0$.
3. Push ℓ into O.
4. Insert the indices of the extrinsic features that involve a relationship with the object of class Y_ℓ into $\mathcal{F}_{\text{kn}}^{(t)}$ to obtain $\mathcal{F}_{\text{kn}}^{(t+1)}$.
5. If $\mathcal{F}_{\text{kn}}^{(t+1)} = \mathcal{F}_{\text{in}} \cup \mathcal{F}_{\text{ex}}$, then stop, otherwise go to step 2.

At the end, O contains the sequence of the binary classifiers.

3.2 Feature Ranking by Cross-Validation

The first method only takes into account the performance of a binary, kNN-like classifier for each class in each feature dimension. The second method is based on cross-validation: a cross-validation is performed at each step of the incremental classification process to select the best binary classifier with respect to the space of currently known features. By doing this, the classification error rate is minimized at each step. The order can be determined with the following steps:

1. $\mathcal{F}_{\text{kn}} = \mathcal{F}_{\text{in}}$. O is an empty FIFO list.
2. Only take into account known feature on the data set : $[x_{ij}]_{i\in\mathcal{F}_{\text{kn}}^{(t)}, 1\leq j\leq N}$.
3. Split the data set into N groups.
4. For each group:
   (a) Use the data from the $N-1$ other groups as a training set and build a model for each class that are not present in O.
   (b) Measure the model performance on the validation set (data from the selected group).
5. Compute $\ell = \max_k p_k$, where p_k is the mean performance for the binary classifier identifying the class Y_k.
6. Push ℓ into O.
7. Insert the indices of the extrinsic features that involve a relationship with the object of class Y_ℓ into $\mathcal{F}_{\text{kn}}^{(t)}$ to obtain $\mathcal{F}_{\text{kn}}^{(t+1)}$.
8. If $\mathcal{F}_{\text{kn}}^{(t+1)} = \mathcal{F}_{\text{in}} \cup \mathcal{F}_{\text{ex}}$, then stop, otherwise go to step 2.

Like in the first method, O contains the sequence of the binary classifiers.

4 Incremental Feature Computation and Classification

The incremental procedure works as follows. First, the centered and normalized training set is stored; $\mathcal{F}_{\text{kn}}$ is initialized at $\mathcal{F}_{\text{in}}$ and the classification order O is computed. Next, the incremental iterations begins:

1. Pop the first element of $O : \ell = \text{pop}(O)$.
2. Train the ℓ^{th} binary classifier on the reduced training set $[x_{ij}]_{i \in \mathcal{F}_{\text{kn}}^{(t)}, 1 \leq j \leq N}$.
3. Give class label Y_ℓ to the objects in the test set having the highest probability to belong this class according to the classifier. At least one super-pixel has to belong to the class Y_ℓ.
4. Compute all extrinsic features that involve a relationship with the object of class Y_ℓ and insert their indices into $\mathcal{F}_{\text{kn}}^{(t)}$ to obtain $\mathcal{F}_{\text{kn}}^{(t+1)}$.
5. If $\mathcal{F}_{\text{kn}}^{(t+1)} = \mathcal{F}_{\text{in}} \cup \mathcal{F}_{\text{ex}}$, then stop, otherwise start a new iteration.

Once a feature order is determined, the classifiers can be trained beforehand to increase the computational efficiency. At the end of the procedure, all features are known, but the classification might not be optimal. Some super-pixel might not be classified, while others can be classified in several classes. A multiclass classifier can address these issues. The multiclass classifier is trained on the whole training set. The multiclass classifier is fed with all features to obtain the final class label for each super-pixel. This last step gives the object their final class label and slightly improves the results, as shown in the experiments.

5 Experiments and Results

As a proof of concept, the principle of incremental classification is illustrated with a simple problem of image segmentation. The data consists of artificial 2D images of crowns and discs encompassing each others, like organs or tissue layers. In each image, there are 8 labels, as shown in Fig. 1. Noise is added to get realistic images. The position, orientation, size, and color of the depicted objects vary in each image. The disc labeled 3 is the only white circle. The discs and crowns with label 4, 6, and 8 are black, while those labeled 1, 2, 5, 7 are gray.

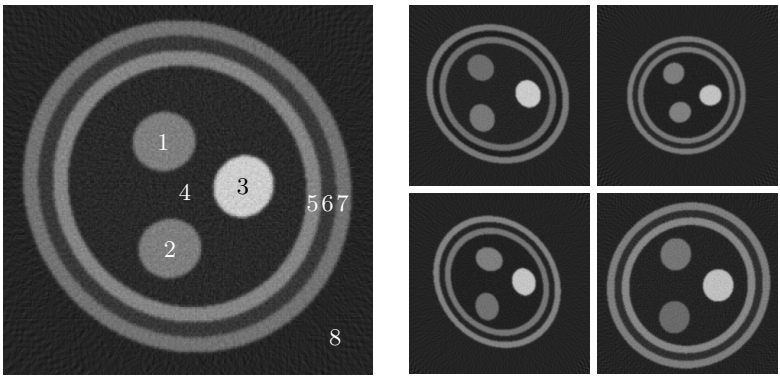

Fig. 1. Left: Image with the position of the 8 different labels. Right: Some images picked in the data set, showing the variations in shape and position.

The data set contains 50 images. Each of them is over-segmented with the watershed-cut algorithm to obtain 20, 30, 40, 50, 75, 100, 125, 150 or 200 super-pixels (see Fig. 2).

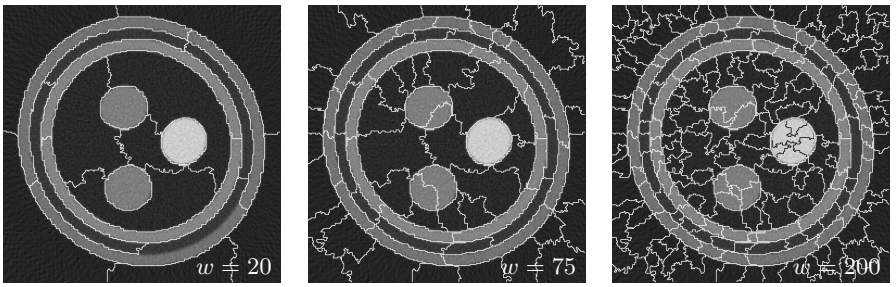

Fig. 2. Exemple of different watershed segmentations. Left: image with 20 super-pixels. Center: 75 super-pixels. Right: 200 super-pixels.

For the data set $\mathbf{X}^w$ (where w is the number of super-pixels), we have $N = 50{*}w$. Six intrinsic features are extracted: the luminance, the mass center coordinates, the height, the width, and a binary feature that indicate if the super-pixel touches the border of the image. Three extrinsic features by class are also computed: the signed distance (or offset) to the center of the class and a binary feature that indicates if the super-pixel is adjacent to the class. At the end, there are 24 extrinsic features. Altogether, there are 30 features.

This data set is randomly split into a training set with 45 images and a test set with the remaining 5 images.. The known features in the test set are centered and normalized by subtraction of the mean and division by the standard deviation computed on the training set. To compute the binary classification order, the whole training set is used for the ranking by nearest neighbors. For the cross-validation method, the training set is split in 10 groups. Each group serves as a validation set during the determination of the order while the rest is used as the training set in the cross-validation process. In both cases, the whole training set is used to build the binary and multiclass models. Our method is implemented with three algorithms: k nearest neighbors (kNN ([6]), support vector machine (SVM [8]) and random forest (RF [4]). For the kNN classifier, we set $k = 3$ (other values give similar results). The SVM uses a Gaussian kernel. The RF grows 500 classification trees. The extrinsic features are inferred for each image individually. During the classification step, if no super-pixel can be identified by the binary classifier, we select the super-pixel that has the highest probability to belong to the class we wish to identify. For the kNN classifier, we choose the super-pixel that has the highest number of neighbors belonging to the considered class. For the SVM, we take the super-pixel with the shortest distance to the classification margin. For the RF, we use the super-pixel that has the highest number of trees classifying it to the considered class. The whole classification procedure is repeated with 20 different training sets.

As the classes are unbalanced, the accuracy is measured with the BCR (balanced classification rate). The BCR is defined as $BCR = \frac{1}{l}\sum_{i=1}^{l}\frac{T_{Y_i}}{|Y_i|}$, where l denotes the number of class, $|Y_i|$ is the number of pixels that should be labeled Y_i and T_{Y_i} is the number of pixels well classified in class Y_i. The BCR is analysed at the end of the incremental procedure (solid line in Fig. 3) and just before the final multiclass classification (dotted line in Fig. 3). The results are compared with an incremental classification where the order is selected randomly (triangles in Fig. 3) and with a baseline classification using only the intrinsic features (gray line in Fig. 3).

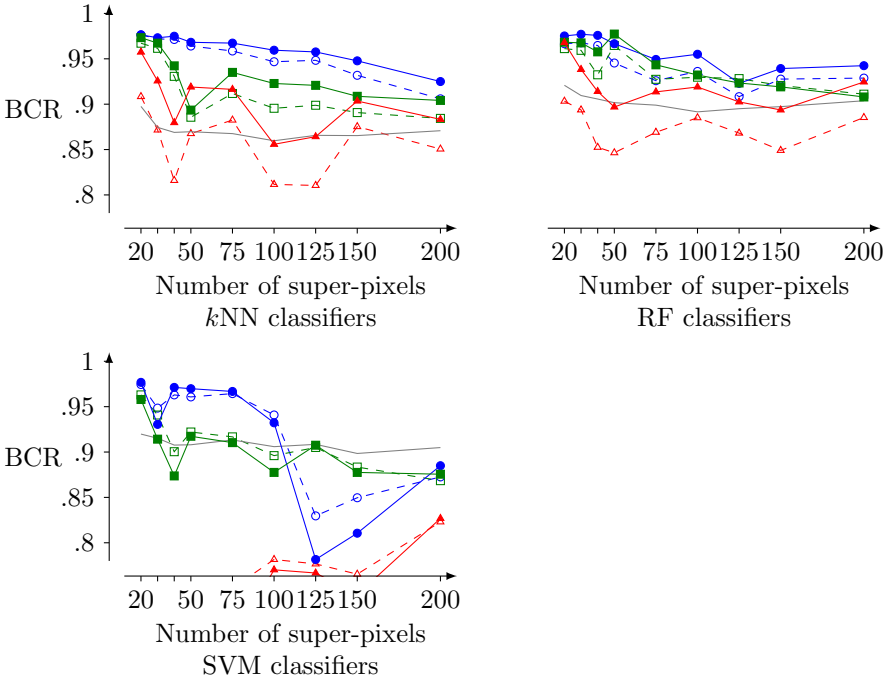

Fig. 3. BCR with respect to the number of super-pixels, with kNN classifiers (upper left), RF classifiers (upper right) and SVM classifiers (bottom). Circles (● and ○) represent the measures with the order by cross-validation. Squares (■ and □) represent the measures with the order by nearest neighbors. Triangles (▲ and △) represent the measures with a random order of extraction. — is the measure while using only the intrinsic features. Filled lines (●, ■ and ▲) are used for the measure after the final multiclass classification. Dashed lines (○, □ and △)are used for the measure before the final multiclass classification.

Figure 3 shows that SVM classifiers perform rather poorly in our incremental method. With SVMs, the multiclass classification decreases the BCR. Moreover, SVMs combined with the nearest neighbors ranking never goes beyond the baseline (classification with intrinsic features only). Eventually, the BCR also falls down when the number of super-pixels increases. The lower BCR can be explained by the fact that during the classification step we have to identify at least one super-pixel belonging to a class at each iteration. If we cannot find one we select the super-pixel with the smallest distance to classification margin. This measure does not identify a good candidate and the features extracted from the classification are wrong. Those mistakes are propagated during the classification.

Although it is the simplest, the kNN classifier yields better results. The final multiclass classification with the kNN improves the BCR. Indeed, at the end of the incremental process with binary classifiers, some super-pixels may remain unlabeled and the final classification addresses this issue. Nevertheless, this last iteration cannot correct all past classification mistakes. As expected, the incremental kNN classifiers pass the baseline and outperform the random feature order. The standard deviation is smaller with the order based on cross-validation (results not shown).

The results with the RF are as good as those with the kNN. The BCR is even more robust when the number of super-pixels increased.

The classification results for the kNN with the different feature ranking methods are shown in Fig. 4.

A detailed analysis of the results, image per image, shows that very often all classifiers make identical errors in the same image of the data set. Errors typically happen in images that depicts the objects in unusual configurations, like when they are the biggest, the smallest, the most shifted, the brightest, etc. Results for those images are naturally poorer, since such configurations are seldom instantiated in the data set.

We also compared the segmentation and labeling results of our incremental classifier to those obtained with atlas registration. For this purpose, we used MIRT (medical image registration toolbox[1]). For each of the 50 images in the data sets, we picked the one that correlates best, among the 49 others. Next, we non-rigidly registered the latter to match the former, using MIRT 2D. The deformation field is parameterized with B-splines, image similarity is measured with mutual information, and registration goes through 5 hierarchical levels with lower-resolution images. These settings allow nonlinear gray-to-gray mappings to be identified, as well as large deformations to be easily captured. Finally, the deformation field was applied to labels associated with the mobile image, in order to determine the labels on the fixed image. The relative simplicity of the images, the choice of the best-correlating image, and the quite powerful settings prevented any convergence failure. The average BCR reached 0.9672. Our methodology based on incremental classification was thus capable of performing better, with BCR values going up to 0.977 in Fig. 3.

[1] `https://sites.google.com/site/myronenko/research/mirt`

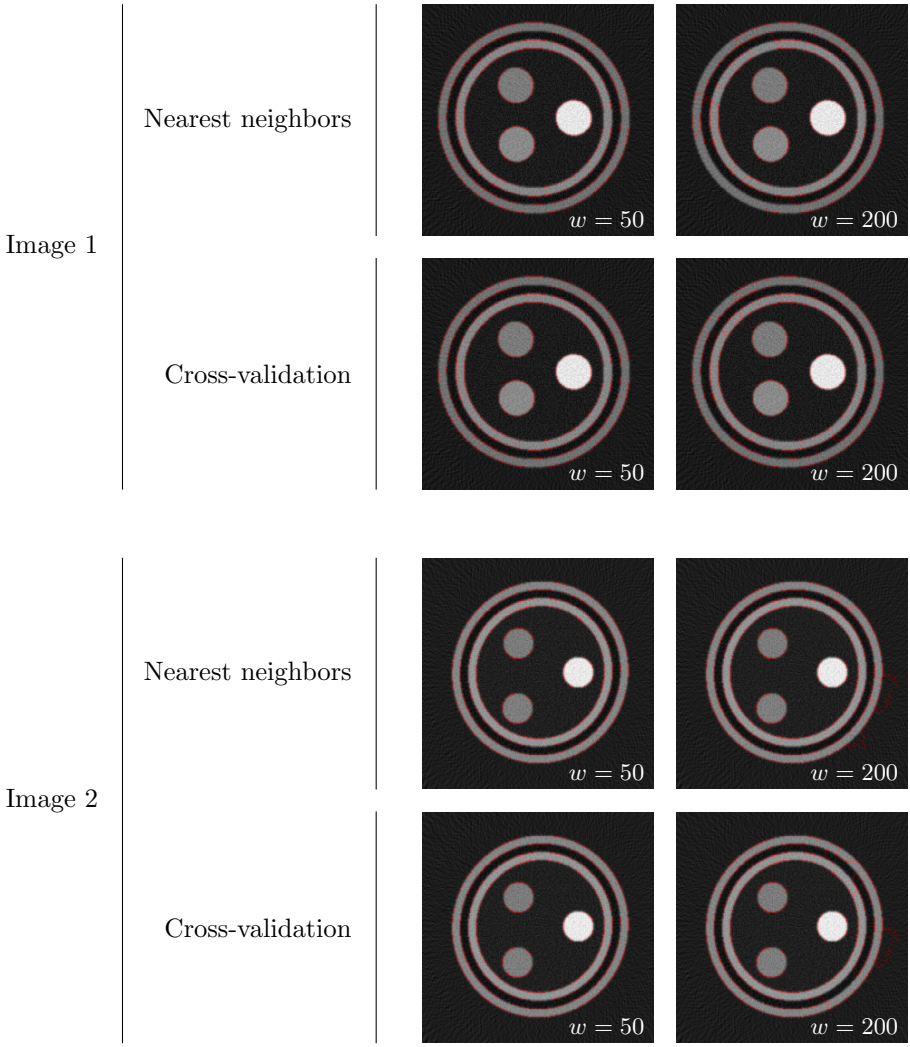

Fig. 4. Result of the final classification with kNN for 2 different images. In each case, the top images are the resulting images with nearest neighbors and the bottom images are the resulting images with cross-validation. The left images are segmented in 50 super-pixels and the right images are segmented in 200 super-pixels.

6 Conclusion

This paper describes a procedure for incremental classification with two methods of sequential feature extraction. It can deal with problems where the value of some features requires a partial classification to be already known. The process of incremental classification aims at refining the partial classification in an iterative way. The procedure is generic and can solve the sub-problems in each iteration with various classification techniques (e.g. naives Bayes, kNN, SVM, decision tree, random forest etc.). The final multi-class classifier can be changed as well. Depending on the problem at hand, the procedure must be adapted with appropriate definitions of features and relevance factors. Failure to do so increases the risk of error propagation in the incremental process. Experiments on artificial images show that the procedure is effective. Nevertheless, the results must be reproduced on real images for a full validation of the method.

In the future, we will investigate the possibility of using a single classifier that deals with all intrinsic and extrinsic features at all times, thanks to the use of adaptive relevance factors.

References

1. Beucher, S., Lantuéjoul, C.: Use of watersheds in contour detection (1979)
2. Beucher, S., Meyer, F.: The morphological approach to segmentation: the watershed transformation. Optical Engineering-New York-Marcel Dekker Incorporated 34, 433–433 (1992)
3. Bondiau, P., Malandain, G., Chanalet, S., Marcy, P., Habrand, J., Fauchon, F., Paquis, P., Courdi, A., Commowick, O., Rutten, I., et al.: Atlas-based automatic segmentation of mr images: validation study on the brainstem in radiotherapy context. International Journal of Radiation Oncology* Biology* Physics 61(1), 289–298 (2005)
4. Breiman, L.: Random forests. Machine Learning 45, 5–32 (2001)
5. Cousty, J., Bertrand, G., Najman, L., Couprie, M.: Watershed cuts: minimum spanning forests and the drop of water principle. IEEE Transactions on Pattern Analysis and Machine Intelligence 31(8), 1362–1374 (2009)
6. Cover, T., Hart, P.: Nearest neighbor pattern classification. IEEE Transactions on Information Theory 13(1), 21–27 (1967)
7. Gould, S., Rodgers, J., Cohen, D., Elidan, G., Koller, D.: Multi-class segmentation with relative location prior. International Journal of Computer Vision 80(3), 300–316 (2008)
8. Joachims, T.: Making large scale svm learning practical (1999)
9. Kass, M., Witkin, A., Terzopoulos, D.: Snakes: Active contour models. International Journal of Computer Vision 1(4), 321–331 (1988)
10. Najman, L., Couprie, M.: Building the component tree in quasi-linear time. IEEE Transactions on Image Processing 15(11), 3531–3539 (2006)
11. Otsu, N.: A threshold selection method from gray-level histograms. Automatica 11, 285–296 (1975)
12. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence 22(8), 888–905 (2000)