Image Categorization based on Segmentation and Region Clustering

Janez BRANK
Jožef Stefan Institute, Jamova 39, Ljubljana, Slovenia

Abstract. The image categorization problem is about assigning images to a set of predefined categories or classes. Machine learning algorithms can be used to learn models that will assign images to classes, but most machine learning algorithms cannot work with images directly. Therefore, we need to introduce a representation of images that typical learning algorithms can use. We propose two image representation methods based on texture segmentation. In the region clustering approach, segmentation is applied to all images in the training set. The descriptions of the resulting regions are then clustered, and images are represented by sparse vectors where each component indicates whether, and to what extent, regions from a particular cluster are present in the image. Algorithms such as Support Vector Machines can then be used to train on the resulting vectors. Alternatively, a similarity measure between segmented images may be taken as a starting point and converted into a generalized kernel for use with Support Vector Machines. We compare these two approaches to a more basic representation based on autocorrelograms.

1. Introduction

Collections of images, also known as pictorial databases, are becoming more and more common in many situations. They include proprietary collections and stock photo databases such as Corbis, photographic collections in libraries and archives, as well as collections of images gathered from the World Wide Web. This gives rise to a need for automated, computer-based ways of managing and using such collections.

1.1 Related work in image retrieval

Most of the work in this area has focused on the problem of *image retrieval*, where the user poses a query to the system and expects the system to retrieve, from the database, images relevant to that query. Several important and interrelated issues need to be addressed in this context: (1) query representation (this must allow the user to adequately express his or her information requirements); (2) image representation (this, in combination with (1), must be of use to the system for determining whether an image is relevant for the query or not); (3) efficiency (the system should be able to handle databases containing thousands of images). Several approaches towards these issues have been proposed, and they may be roughly divided into textual, semantic, and visual image retrieval.

The textual approach consists of describing each image with a set of keywords, and the user's query, which is likewise a set of keywords, is interpreted as a requirement to retrieve

images whose descriptions contain some (or all) of the query keywords. This is relatively straightforward and can be implemented efficiently, but it suffers from problems such as polysemy (the same word means several things, especially to different people; also, different people may choose different words to describe the same scene), as well as from the need to provide the textual descriptions for images. The costs of assigning descriptions manually are likely excessive if the collection is large. The result is that this approach has not received as much attention in the research literature, but it has been very popular in practical settings where reasonably good textual descriptions can be obtained automatically. For example, Google and AltaVista provide image search capabilities where keywords for an image have been obtained from its URLs and from the HTML document referencing that image.

The semantic approach is similar, except that the descriptions are generally structured, e.g. organized in semantic frames, and that efforts may be made to identify different words for the same concept. This tends to require even more manual attention and makes automatic preparation of descriptions practically impossible. Thus, for the present, this approach is generally not viable in practice.

The visual approach, also known as *content-based image retrieval*, has received the most attention in the literature. Here the query is itself an image, which the user points out or provides in some other way (possibly by sketching), and the system is supposed to retrieve images that "look similar" to the query image. This still leaves open the question of image representation, and of measuring similarity between two images.

Numerous representations and similarity or distance measures have been proposed. One popular approach is to partition the colorspace (i.e. the space of all possible colors which a pixel in the image may have) into some number of disjoint buckets, then treating colors from the same bucket as indistinguishable. This "quantization" of the colorspace is usually fixed in advance and is constant for all images, including query images. An image can then be described by considering each color in the quantized colorspace and expressing, with one or more numerical quantities, the relation of that color to the image. If we store the proportion of the image covered by each color, the resulting vector is called a *histogram*. Histograms are an old and well-known way of describing images [1], but their deficiency is in that they discard all information about the spatial distribution of colors in the image, recording only the amounts in which particular colors are present. Various improvements have been proposed to address this problem. For example, *autocorrelograms* consist of probabilities, for each color c and for several distances d, that a pixel, chosen at random at distance d from a randomly-chosen pixel of color c, is itself of color c. Such probabilities capture come information both about the extent to which a color is present in the image, and about its distribution in space. Huang *et al.* [2] propose taking e.g. $d = 1, 3, 5, 7$. A distance measure between two images can then be defined using a vector norm measuring distance between their histograms or autocorrelograms; the Manhattan norm has been generally found preferable to the Euclidean for this purpose, although other distance measures have also been proposed.

Describing every image with a single vector of a fixed size may well be considered an important deficiency; after all, an image likely consists of several distinct regions, such as various objects and the background. Since one cannot, for a general collection of images, realistically hope to automatically extract objects as perceived by the user, one can at least try to separate the image into regions such that each is relatively homogeneous in appearance (i.e. in color and/or texture), and distinct from adjacent regions. This process is known as (*texture*) *segmentation*, and is still a subject of ongoing research in the fields of computer vision and information retrieval. For image retrieval, each region of a segmented image would be described separately, and some measure of similarity (or difference) between regions introduced; when comparing two images, similarities between their regions must be somehow combined into a single similarity measure between the two images.

Various approaches to texture segmentation are known, but for the needs of image retrieval the emphasis must be rather on efficiency (which allows one to process many images quickly) than on detecting exact borders between regions. One way of achieving this is to divide the image into a grid of small tiles or *windows* (e.g. 4 by 4 pixels), describing each window by a vector of a fixed size (average colors, or the main coefficients from two-dimensional discrete wavelet transforms, are commonly used for this purpose), then clustering the window descriptions. The assumption here is that windows whose descriptions lie close together in their vector space must be roughly similar in appearance, and it would therefore make sense to make such windows part of the same region. Thus one region is defined for each resulting cluster, and the centroid of the cluster is used as a description of the region. Algorithms used for clustering in this phase include BIRCH [3], used in [4], and *k*-means, used in [5]. Our previous experience with image retrieval suggests that *k*-means is lacking in robustness in this situation, with the resulting partition too much dependent on the initial placement of centroids. BIRCH builds a hierarchy of clusters by inserting the data points into a tree structure (CF-tree), where inserting a point into the tree is guided by the efficiently aggregated information about subtrees that is stored in each node.

To measure similarity between two segmented images, Natsev et al. [4] proposed a query-based paradigm, where all pairs of regions (one region from each image) are found such that the difference between the vectors describing the two regions is below a certain threshold; one then computes what part of the total area of the two images is covered by the union of all these regions, and uses this ratio as a similarity measure between images. A more sophisticated, though also more time-consuming, approach has been proposed by Li et al. [6], called *integrated region matching*. There, a distance is computed for all pairs of regions, and a weighted sum of these distances is used as a distance between the two images. Weights are assigned so as to prevent a region from gaining an influence out of proportion to its size (area), and priority is given to pairs that are more similar.

1.2 Image categorization

In the problem of image categorization, we assume that a fixed set of classes or *categories* has been defined, and that each image belongs to one category. We are given a set of training images, whose category membership is known. In addition, we have a set of unlabeled images, and we want to assign each of these images into one of the categories.

For an automatic approach to this problem to be feasible, one needs to assume that images that are in a way similar will fall into the same category, and that this underlying notion of similarity can be captured automatically using a suitable representation of images and some learning algorithm. In addition, before applying machine learning (ML) techniques to images in order to learn models (also known as classifiers or predictors), we need to represent images with structures that known machine learning algorithms can actually work with. Thus the problem of image categorization bears a strong relationship to image retrieval, where the notions of representation and similarity are also of great importance.

One way of applying results from image retrieval to the problem of image categorization would be to take simple vector-based descriptions of images, such as histograms or autocorrelograms, and use any of the numerous ML techniques that can work with vectors. Alternatively, one could take an arbitrary similarity measure from image retrieval, no matter what the underlying representation, and use it in combination with a ML algorithm that can work with arbitrary similarity or distance measures, such as nearest-neighbors (also known as instance-based learning), or possibly the generalized kernel variety of support vector machines [7].

1.3 Support vector machines

The support vector machine (SVM) [8, 9] is a well-known family of algorithms that has gained a wide recognition in the recent years as one of the state-of-the-art machine learning algorithms. In addition to classification, versions of this algorithm support other tasks, such as regression, principal component analysis, or estimating the support of a probability distribution.

The basic formulation of the SVM for a two-class classification problem assumes that the training examples are vectors, and that members of one class should be separated from those of the other class by a hyperplane. The hyperplane should be chosen so that the examples nearest to it are as far away as possible; in other words, one should maximize the *margin* between the hyperplane and the training examples. This is both intuitively appealing and has a foundation in the statistical learning theory. In addition, one usually permits some training examples to be misclassified, thereby allowing the algorithm to look for a tradeoff between a large margin on the one hand and high accuracy on the training set (at the risk of overfitting) on the other hand.

Furthermore, the optimization problem on which the SVM is based can be converted to its dual form, where training vectors need not be handled explicitly, as long as we can compute dot products between them. This allows one to use the so-called "kernel trick", which consists of mapping the training vectors into a higher-dimensional space in such a way that allows dot products of their images under this mapping to be computed efficiently without ever storing these images explicitly. The function which calculates such dot products is called a *kernel*.

While the original formulation of the SVM assumes a two-class problem, several strategies can be used to work with multiclass problems as well, by transforming them into groups of two-class problems. For example, in the one-against-one approach, we train, for each pair of classes, a classifier that separates instances of one class from those of the other; for classification, all the classifiers are shown the instance in question, and vote for one or the other class; in the end, we choose the class that received the greatest number of votes. Hsu and Lin [10] found that this method compares favorably with several other approaches to converting multiclass problems into binary ones.

2. Using segmentation for image categorization

It seems plausible that image segmentation should be able to uncover useful information about the contents and appearance of an image, information that should be helpful for categorizing images. Segmentation has certainly done well in image retrieval, as shown by its frequent use in the literature of that field. We considered several ways of combining segmentation with machine learning techniques, and will now proceed to describe them.

2.1 The nearest-neighbor method

Here we take a similarity measure between segmented images, such as the WALRUS or the IRM measure, and combine this with the nearest-neighbor method to produce a classification function for unseen images. The nearest-neighbor algorithm, given a query image I_q, looks for the most similar training image (called the "nearest neighbor" of I_q) and predicts that I_q belongs to the same class as its nearest neighbor. Alternatively, the algorithm may consider

several near neighbors, which then vote for their respective classes, possibly with each vote being weighted by how similar the neighbor in question is to the query image. The important property of the nearest-neighbor approach is that it doesn't make any strong assumptions about the similarity measure we want to use with it.

However, there are numerous machine learning algorithms that can do better than nearest-neighbor in many situations, but they make more assumptions about the nature of the instances they are working with. For example, the support vector machine (SVM) has been found to be very successful in numerous domains, but it is designed for learning on vectors and particularly assumes that dot products can be computed between the instances. Thus, it can certainly be used for image categorization if images are described by vectors such as histograms or autocorrelograms. Segmentation, on the other hand, represents the image by a set of regions and each region by a vector. We consider the two approaches of combining segmentation with the SVM algorithm: generalized kernels and region clustering.

2.2 Generalized kernels

The SVM algorithm assumes that the training examples (in our case, the training images I_j) can be mapped using some mapping j into a (possibly high-dimensional) vector space F, often called the "feature space", and that dot products between the resulting vectors can be effectively computed. In other words, we need to be able to evaluate the function $K(I, I') = \langle j(I), j(I') \rangle_F$ for every pair of training examples I and I' (here, $\langle \cdot, \cdot \rangle_F$ is the dot product in the feature space F). There is no need to compute the vectors $j(I)$ explicitly unless this is necessary in order to determine the value of K.

A kernel is in some ways similar to a similarity measure, because it has larger values when the vectors involved have similar directions, just like a similarity measure has larger values when the items being compared are more similar. However, not any similarity measure can be directly used as a kernel, because a kernel has to correspond to a dot product in some vector space F, which is not necessarily true for any similarity measure. Using a non-dot-product function as a kernel means we no longer have guarantees that the SVM training algorithm will converge, and even if it does converge we no longer have any theoretical grounds to expect good classification performance from the resulting model.

To address these issues, Mangasarian [7] proposed the notion of a *generalized kernel*, whereby an arbitrary function of two training examples, e.g. $K(x, x')$ can be incorporated in an optimization problem similar to the one used in the usual formulation of the support vector machine. Given a set of training examples, x_i, and their labels, y_i (where $y_i = +1$ signifies a positive example and $y_i = -1$ a negative example), the original SVM formulation would try to separate positive from negative instances using a discrimination surface of the form $\langle w, j(x) \rangle_F = 0$, where $w = \Sigma_i \alpha_i y_i j(x_i)$. The α_i coefficients would be obtained by solving an optimization problem that minimizes $\|w\|_F^2$ (subject to the constraint that the training examples lie on the correct side of the discrimination surface), thereby maximizing the margin (the distance from the discrimination surface to the nearest training example in the F-space). The discrimination surface can then also be written as $\Sigma_i \alpha_i y_i \langle j(x_i), j(x) \rangle_F = 0$, and the dot product can be computed using a kernel function: $\Sigma_i \alpha_i y_i K(x_i, x) = 0$. Now suppose we have a function K that is not a proper kernel; we can nevertheless use it in a decision surface of this form. However, for such a K there might not exist an F, $w \in F$, and j such that $\Sigma_i \alpha_i y_i K(x_i, x)$ corresponds to $\langle w, j(x) \rangle$. Thus, we cannot minimize $\|w\|_F^2$; instead, we can minimize $\Sigma_i \alpha_i$, which can informally be interpreted as striving towards a simpler discrimination surface and trying to make use of as few support vectors as possible.

It can be shown that this formulation is equivalent to using n-dimensional real space as the

feature space (i.e. $F = R^n$), where n is the number of training examples, and mapping an instance x into the vector $j(x) := (K(x_i, x))_i$. The dot product in this space is equivalent to $\langle x, x' \rangle_F = \Sigma_i K(x_i, x) K(x_i, x')$. Thus, if K is some similarity measure between instances (images in our case), the generalized kernel approach can be seen as representing each example by a vector $(K(x_i, x))_i$ of the similarities of this example to all the training example x_i. In other words, two images I and I' will have a large value $\langle I, I' \rangle_F$ (implying they are similar) if they both exhibit a similar pattern of similarities to training images. As above, note that we did not need to make any assumptions concerning the properties of the similarity measure K. Thus, the generalized kernel approach allows us to combine any image similarity measure with SVM as the training algorithm for the purposes of image categorization.

2.3 Region clustering

The regions obtained after the segmentation of an image can be seen a description of that image; however, different images will give rise to different and unrelated regions, making comparison of several images difficult. Thus it seems promising to bring the descriptions of all images to a "common denominator", making it easier to work with an entire set of images as a whole.

Assuming we have a vector describing each region of each image (which is true when segmentation algorithms such as WALRUS are used), we can take all the regions of all the training images, and run a clustering algorithm over the set of vectors describing all these regions. Regions with similar appearance or structure, regardless of whether they originated in different images or not, will tend to be described by similar vectors, which would therefore be expected to fall into the same cluster. Thus, each resulting cluster can be interpreted as a group of similar regions, probably appearing in several different images. Then, to describe some particular image, we would record which region groups are represented on this image; in other words, for each region group, what percentage of the image is covered by regions from that group. For an image not included in this initial clustering process, we could still find, for each region R from that image, the region group with the nearest centroid and consider R as a representative of that region group. Each image is now represented by a vector with as many components as there are region groups, and these vectors can be used as an input into the SVM or many other algorithms.

In other words, this approach builds, after examining the entire image collection, a "feature space" where each feature corresponds to a group of regions. The mapping which maps an image to its representation under this feature space does not really depend on that image alone but on the entire collection of images from which the region groups have been built.

3. Experiments

3.1 Design of the experiments

So far, no standard datasets have emerged in the fields of image retrieval and image categorization. Different authors work with different collections that are generally not publicly available. We used the *misc* database from *http://www-db.stanford.edu/IMAGE/* to compare different image categorization strategies. This collection consists of about ten thousand images, basically photographs of very diverse scenes and objects. Thus it is quite

suitable for such experiments, being large enough to be diverse, but not so large as to become unmanageable. It has also been used in image retrieval before, by Wang *et al.* [11] and Natsev *et al.* [4]. The images in this collection are not labelled or categorized, however. By manual inspection of the image set, we formed a group of 14 categories covering a total of 1172 images; the remaining images were not used in the experiments reported in this section. The categories were chosen with the intention of having categories of various sizes and various levels of difficulty. Some categories are quite distinct in their typical color structure, while others are much more similar in this way and should pose a greater challenge to machine learning algorithms (e.g. sea, clouds, and mountains, which all have a predominance of white and blue tones). The names of the categories are (the number of images in each category is given in parentheses): butterflies (101 images), the US flag (32), sunsets (224), autumn (68), flowers (201), planets (56), satellite images of Earth (61), cars (99), mountains (60), clouds (65), sea (35), surfboards (71), sailboats (37), prairie animals (62). A list of all images used can be found at: *http://quintus.ijs.si/janez/dip/splits.html.*

We used the LibSvm [12] program to train SVM classifiers. It natively supports multiclass problems using the one-against-one strategy [10]. For the transductive learning experiments, we used the SvmLight program (version 3.50) by Thorsten Joachims [13], which contains an implementation of Joachims' transduction algorithm for SVM [14]. We use linear kernels throughout these experiments, except in the experiments focusing on the generalized-kernel approach; our preliminary experiments suggested that switching to nonlinear kernels does not generally bring about a significant improvement in classification accuracy.

3.2 Results and discussion

We compared the following approaches to learning classifiers:

- Autocorrelograms, based on the HSV (hue, saturation, value) colorspace, quantized into 256 buckets by splitting the H axis into sixteen and the other two into four equal parts.
- Generalized kernels for SVM, based on the WALRUS segmentation and the IRM similarity metric. (Preliminary experiments have shown IRM to be much more promising than the simpler similarity metric proposed by the authors of WALRUS; this is why IRM was used in the experiments presented here.)
- Region clustering, as described in 2.3 above. The same segmentation was used as for the experiments on generalized kernels. To cluster the descriptions of regions over all training images, we used the same BIRCH algorithm that is also used for clustering during the segmentation of individual images. SVM was then used to train on the image descriptions based on region clustering.

Stratified ten-fold cross-validation was performed in all cases. Table 1 shows the average and the standard error of the classification accuracy for each of these experiments. For comparison, we also performed an experiment using the nearest-neighbor method and the IRM similarity metric; it achieved a classification accuracy of 69.1 % ± 1.3 %.

Table 1. Classification accuracies of classifiers based on different image representations. See text for details.

Method description	Classification accuracy
Autocorrelograms	80.2 % ± 1.3 %
Generalized kernels	79.0 % ± 1.3 %
Region clustering	70.0 % ± 1.6 %

The result that autocorrelograms perform as well or better than the two segmentation-based approaches is surprising (the differences in accuracy between autocorrelograms and generalized kernels are in fact not statistically significant, but it is fair to also take into account the greater computational complexity of the generalized kernel approach). A closer examination of the clustering phrase of the region clustering approach suggested that the clustering is highly unstable, meaning that, for example, if the centroids of the resulting clusters are stored and all regions redistributed to the clusters with the nearest centroids, many regions will move to a different cluster. This would mean, for instance, that two very similar regions in different images might fall into different clusters; in addition, the resulting set of clusters is quite sensitive to the order in which the images were considered. The authors of BIRCH [3] propose several redistribution passes to increase the stability and quality of the resulting clusters. In our experiments, even after 5 or 10 such redistribution passes, many regions were still migrating from one cluster to another. A more stable approach to clustering might improve the performance of this approach, although our preliminary experiments with a graph-based clustering (each region becomes a vertex, and an edge is formed where two regions are sufficiently similar; then each connected component can be interpreted as defining a cluster) were not promising.

An alternative way of addressing the problem of cluster stability and the compatibility of the representations of different images would be to involve all the images, both from training and the test sets, in a common clustering. This amounts to a form of transduction, or treating test data as additional unlabelled training data. With this provision, the region clustering approach achieved an accuracy of 86.4 % ± 1.0 %. However, for the comparison to be fair, we should also include transduction in the SVM training process, where it can be combined with autocorrelograms or other methods as well. Since LibSvm does not support transduction, we used SvmLight in this part of the experiments. It implements Joachims' transduction SVM approach [14], which consists of assigning labels to the unlabelled instances in such a way as to maximize the margin of the classifier obtained after training on the union of these and the ordinary training instances. With transduction SVM, region clustering achieves accuracies around 91.9 % ± 1.0 %, while autocorrelograms achieve accuracies around 90.7 % ± 1.1 %. A t-test shows the difference to be significant with a confidence level of 0.945, i.e. slightly below the customary 0.95.

4. Conclusions and future work

Our experiments show that it is difficult to use segmentation-based image representation methods in image categorization. A straightforward way of using segmentation, such as the nearest-neighbor method combined with the IRM metric, has actually been found to perform much worse than conceptually much simpler global image representations such as autocorrelograms (which are also simpler and more efficient to compute). More sophisticated ways of using information obtained from segmentation, such as the generalized kernel approach and (to a lesser extent) the region clustering approach, can compete with autocorrelograms but cannot significantly outperform them.

We nonetheless believe that there must be ways of using segmentation profitably for image categorization, just as it is used in image retrieval, and that this is still an interesting topic for future work. In particular, it would be interesting to further explore the influence of the clustering algorithm used in the region clustering approach, and to look for more

stable clustering algorithms that would allow the region clustering approach to perform better in the inductive in additional to the transductive setting.

Furthermore, segmentation is a relatively complex task, and segmentation algorithms usually depend on several parameters, such as (in the case of a clustering-based approach to segmentation) sizes of windows into which the image is divided before segmentation, the descriptions of individual windows, the clustering method (and its parameters) used to cluster the windows into regions. One might want to explore the influence of these various parameters on the segmentation (and subsequently on the suitability of the resulting representation for image categorization) in a more systematic way.

Additionally, the almost equally good results achieved by the best classifiers based on different representations suggest that this may be simply the best reasonably achievable accuracy for this particular dataset, and that the remaining errors are due to the fact that the present algorithms simply cannot capture some of the "semantic" motivation underlying the category memberships of some images. To verify this hypothesis, it would be interesting to test these approaches on one or more other collections of images.

References

[1] M. J. Swain, D. H. Ballard: *Color indexing*. Int. Journal of Computer Vision, 7(1):11–32, November 1991

[2] J. Huang, S. R. Kumar, M. Mitra: *Combining supervised learning with color correlograms for content-based image retrieval*. Proc. 5th ACM Multimedia Conference, Seattle, USA, 1997, pp. 325–334.

[3] T. Zhang, R. Ramakrishnan, M. Livny: *BIRCH: An efficient data clustering method for very large databases*. Proc. ACM SIGMOD Int. Conf. on Management of Data, Montreal, 1996. pp. 103–114.

[4] A. Natsev, R. Rastogi, K. Shim: *WALRUS: a similarity retrieval algorithm for large databases*. Proc. ACM SIGMOD Int. Conf. on Management of Data, Philadelphia, USA, 1999, pp. 395–406.

[5] J. Z. Wang, J. Li, G. Wiederhold: *SIMPLIcity: Semantics-sensitive integrated matching for picture libraries*. Advances in Visual Information Systems, 4th Int. Conf., Lyon, France, 2000, pp. 360–371.

[6] J. Li, J. Z. Wang, G. Wiederhold: *IRM: Integrated region matching for image retrieval*. Proc. 8th ACM Multimedia Conference, Los Angeles, USA, 2000, pp. 147–156.

[7] O. L. Mangasarian: *Generalized support vector machines*. In: A. J. Smola, P. J. Bartlett, B. Schölkopf, D. Schuurmans (eds.), Advances in Large Margin Classifiers, MIT Press, 2000, pp. 135–146.

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

[9] C. Burges: *A tutorial on support vector machines for pattern recognition*. Data Mining and Knowledge Discovery, 2(2):121–167, June 1998.

[10] C.-W. Hsu, C.-J. Lin: *A comparison of methods for multi-class support vector machines*. Dept. of Computer Science and Information Engineering, National Taiwan University, April 2001.

[11] J. Z. Wang, G. Wiederhold, O. Firschein, S. X. Wei: *Content-based image indexing using Daubechies' wavelets*. International Journal of Digital Libraries, 1(4):311–328, December 1997.

[12] C.-C. Chang, C.-J. Lin: *LibSVM: a library for support vector machines* (version 2.3). Dept. of Computer Science and Information Engineering, National Taiwan University, April 2001.

[13] T. Joachims: *Making large-scale SVM learning practical*. In: B. Schölkopf, C. Burges, A. Smola (eds.), Advances in Kernel Methods — Support Vector Learning. MIT Press, 1999, pp. 169–184.

[14] T. Joachims: *Transductive inference for text classification using support vector machines*. Proc. 16th International Conference on Machine Learning, Bled, Slovenia, 1999, pp. 200–209.