Image Classification with a Novel Semantic Linear-time Graph Kernel

Hongchuan Luo, Zhixing Huang, Guoqiang Xiao *

College of Computer and Information Science,Southwest University
Chongqing, China 400715
*Corresponding author: gqxiao@swu.edu.cn

***Abstract*— Image classification is currently a vital and challenging topic in computer vision. Although it has been achieved many classification algorithms so far, the classification of natural images still remains great difficulties in image processing. In this paper, we propose a semantic linear-time graph kernel for image classification. Each image is represented by a graph and the vertex of each graph corresponds to a segmented region. The semantic graph kernel can be used to compute similarities of graphs efficiently. The performance of the proposed graph kernel is demonstrated via a comparative performance assessment which is carried out on two large natural image databases. Experimental results show both the efficiency and precision of the proposed algorithm are better than those of histogram-based image classification which are carried out on Support Vector Machines (SVMs).**

I. Introduction

Image classification is a fundamental issue in computer vision. It is prevalent in many fields such as image retrieval and object recognition. Although image classification has been studied for years, it is still a difficult problem within digital media. Besides, most classification methods require a pre-processing step by computing color histogram or feature extraction. However, with the development of advanced equipment such as raw camera, images are usually high resolutions, which makes pre-processing more difficult.

Recently researchers usually handle image classification tasks by designing graph kernels which are inspired by natural language process. Videlicet, a kernel-based method, has received an increasingly attention[1]. As is known, graph kernel is an attractive technique to measure the similarity between graphs. In[2], Harchaoui and Bach proposed a family of graph kernels which achieves image classification with a satisfactory result. A graph kernel for semantic link network is proposed in literature[3], which performed a multi-class classification task of documents. Although these graph kernels are suitable for image classification, the time complexity is still high. So in this paper we propose a semantic linear-time graph kernel for image classification. The main idea of this work is to overcome the time complexity deficiency of the conventional graph kernels and achieve image classification within linear-time.

The paper is structured as follows. In Section II, we review related work on graph matching for image processing as well as on kernels for images. In Section III, we present the segmentation algorithm that has been used in this paper. Section IV presents the proposed semantic linear-time graph kernel, which is used to compute the similarities between graphs. Experimental results are shown in Section V and conclusions are given in Section VI.

II. Related work

A graph is an efficient approach to represent structured data, especially its expressiveness and convenience for complex data. Graph matching (GM) plays an important role in computer vision.Recently, more and more graph kernels have been proposed to solve graph matching problems. Literature[2] lists a family of kernels which achieve image classification in an efficient way. In[4], a kernel called Weisfeiler-Lehman Graph Kernel was proposed with its reliable performance. Although these graph kernels can achieve satisfactory results, time complexity of these graph kernels is very high. To this end, we proposed a semantic linear-time graph kernel in this paper, which fulfills graph matching within linear-time. The semantic graph kernel especially can be applied to large graphs in a parallel way and successfully avoid NP-hard Problems.

In image representation stage, images are segmented into regions, which are regarded as graph nodes. If two regions are adjacent, they are considered as neighborhoods. We simply consider the semantic information as weak and strong between neighborhoods. Its implementation will be discussed in detail in the experimental section.

III. Graph cuts segmentation and graphs construction

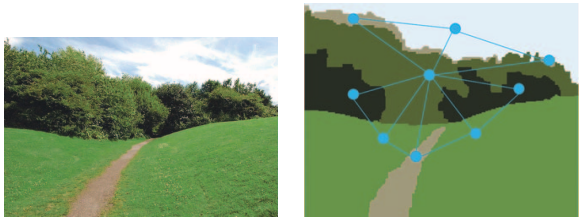
There exists many methods available for the segmentation of natural images[5][6]. In this paper we choose the graph cuts method, which segments image into a given region number and obtains expected segmentation results.

A. Image segmentation with graph cuts

This step is generally essential because it provides a simpler portrayal of the input image.For a given color image, we mapped it into a high dimension feature space via a (Radial Basis Function)RBF kernel function. The high dimension image data which is obtained by RBF often are referred as transformed image data[5]. With the transformed image data, we can construct a graph corresponding to the image and the graph cuts formulation in[5] is applicable. In[5] and[6], Salah and Aye defined the constraints, which are

978-1-4673-9808-4/15 $31.00 © 2015 IEEE
DOI 10.1109/SKG.2015.16

used to evaluate conformity between transformed image data and original image data, as data term. They also optimize an object function which contains two characteristic terms: a data term which is used to evaluate the deviation of the transformed image data and a regularization term which is used to measure the smoothness of the boundary of each region. In this paper, we use graph cuts method in[5] to cut the graph into several subgraphs and use energy function in[7] to obtain the optimized segmentation regions. An example of segmentation is shown in Fig. 1(b).



(a) Original image (b) Segmentation Regions

Fig. 1. An example of natural image(left),and its segmentation region-s(right)obtained by graph cuts.and the graph is shown at the right

B. Graph representation

In the rest of this paper, all the images will be represented by undirected planar graphs. Since images are segmented into several regions, which are represented by nodes in the graph, the degree (number of neighbors) of each node is usually small. For edges, if two regions are adjacent, they are considered as neighborhoods and will be joined by edges. An example of graph representation is shown in Fig. 1(b).

IV. Linear-time Graph Kernel

The general architecture of this work is depicted in Fig. 2. It consists of four major parts:

1) Segment image into regions and construct each image into graph. Regions are represented by nodes and semantic information is represented by edges.
2) Encode graph nodes. Each node of the graph is characterized as a bit array of fixed length.
3) Calculate hash values. Neighborhood hash function is defined and hash values of each node are calculated in the graph.
4) Build graph kernel. Similarities are computed for all the graphs.

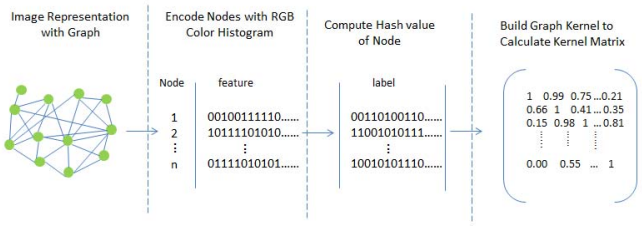


Fig. 2. The process of calculating graph similarity

A. Encoding for graph nodes

As stated in Section II, we assign label for each region by computing color histogram and encode the color histogram into bit array in this paper. In the field of image process each R,G and B component is ranged from 0 to 255, which are usually mapped to 16-level. In order to obtain the bit array of each region, we encode graph nodes with the following three steps:

1) Computing R,G and B color histogram respectively of each node (region);
2) Mapping each R,G and B component to 16-level and yielding three tuple as (R,G,B);
3) Encoding graph nodes by mapping three tuple into 4096*1 dimensional vector.

Finally the code of each node is obtained as in Fig. 2.

B. Neighborhood Hash Operation

Neighborhood hash operation aims at learning the distribution of neighboring nodes. Here we construct the neighborhood hash function. Different semantic definitions of links and distribution of neighbors are reflected by computing different hash values using discriminative representations.

Firstly, we define some basic operations on bit labels in TableI. $L(i) = \{b_1, b_2, ..., b_N\}$ represents corresponding bit label of region i. Bitwise OR, AND and XOR operations between two bit labels $L(i)$ and $L(j)$ will result in another bit label of the same length. Operation ROT will shift the last $(N - x)$ bits to the left by x bits and moves the first x bits to the right end. Operation ADD signifies addition of two bit labels with carries.

TABLE I
Definition of Bit Operations

$OR(L(i), L(j)) = L(i) \mid L(j)$
$AND(L(i), L(j)) = L(i) \ \& \ L(j)$
$XOR(L(i), L(j)) = L(i) \oplus L(j)$
$ADD(L(i), L(j)) = L(i) + L(j)$
$ROT_x(L(i)) = \{s_{1+x}, s_{2+x}, ..., S_N, s_1, s_2, ..., s_x\}$

Given a graph $G =< N, E >$, a semantic node n and its q neighbors Adj $(n) = \{n_1, ..., n_q\}$, the semantic relationship between two neighbors are symbolized by r, and the frequency of a node is defined as the number of regions which have the same color in a graph. Then, hash code of neighboring nodes n_i are computed as:

$$H(n_i) = ROT((L(n)|L(n_i)) + c_i), if\ r\ is\ weak \quad (1)$$

$$H(n_i) = ROT((L(n)\&L(n_i)) + c_i), if\ r\ is\ strong \quad (2)$$

where c_i stands for the frequency of node n_i.

One should try to lower the probability of accidental hash collisions when designing a hash function. For the purpose of avoid hash collision, we also employ ROT and ADD operations.

Rotation operation can distinguish the direction from the source node to the destination node. If the directed edge between node n and its neighbor n_i points to node n itself, the edge is defined as an in-coming edge, otherwise the edge is defined as an out-going edge. Based on the definition , if r is an out-going edge for node n, namely,$n \xrightarrow{r} n_i$, the ROT in the above equation is set to ROT_2. If it is an in-coming edge as $n_i \xrightarrow{r} n$, set ROT to be ROT_3. Sometimes $(L(n)+c)$ overflows when there is a carry on the most significant bit. Omit this carry to keep the length of bit label in consistent.

For undirected graph, the hash code of a neighboring node n_i will be computed as:

$$H(n_i) = (L(n)|L(n_i)) + c_i, if\ r\ is\ weak \quad (3)$$

$$H(n_i) = (L(n)\&L(n_i)) + c_i, if\ r\ is\ strong \quad (4)$$

An example is shown in Fig. 3. There are four neighbors for node R3. The bit label of R1 is 0010010 and its frequency is 0000110. The link from R3 to R1 is an out-going and strong edge. Therefore, the hash code of this neighbor node is calculated as 1011000.

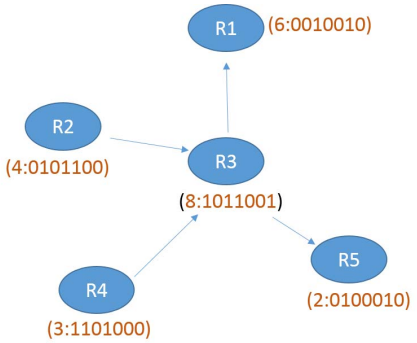


Fig. 3. Example of computing hash value for nodes

Based on the method which is used to calculate hash code of neighbors, the neighborhood hash function for a semantic node is defined as follows:

Given a graph $G =< N, E >$, a semantic node n and its q neighbors $Adj(n) = \{n1, ..., nq\}$. Let $NH_{Adj(n)} = \{H(n_1), ..., H(n_q)\}$ represents the hash code of neighboring nodes. The neighborhood hash code of node n is computed as:

$$NH(n) = (L(n) + c) \oplus (H(n_1) \oplus ... \oplus H(n_q)) \quad (5)$$

where c denotes the frequency of n.

If $(L(n) + c)$ overflows when there is a carry on the last bit, leave out this carry to keep the length of bit arrays in consistent. Computing hash function aims at embedding semantic information on the link into bits array labels. Bit operations can efficiently aggregate the topology information of neighboring nodes, and neighborhood hash features can effectively represent the syntactical structural information of graph.

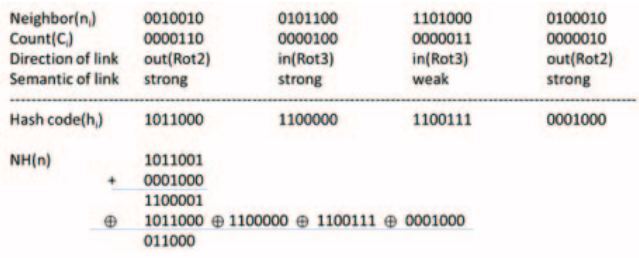


Fig. 4. Example of computing hash value for nodes

Fig. 4 illustrates the process of computing the neighborhood hash value for a semantic node; the semantic definitions of links are consistent with those in Fig.3. The nodes are labeled using the form of (frequency: code), where frequency signifies the frequency of a region and code represents the corresponding bit label of each region. According to those definitions, the bit label 1010111 in the lower right of this figure is the final hash value of node R3. The neighborhood hash operation can be applied iteratively on the node.

The neighborhood hash code which is considered in Equation 5 can represent $1-hop$ structural information of node n. Alternatively, let $NH^r(n)$ represent the neighborhood hash code of node n for $r-hop$ neighbors. It has the ability to represent structural information in the subgraph of radius γ. The revised hash function is given by:

$$NH^r(n) = NH(NH^{r-1}(n)), \gamma > 1 \quad (6)$$

$$NH^1(n) = (L(n) + c) \oplus (H(n_1) \oplus ... \oplus H(n_q)), \gamma = 1 \quad (7)$$

Based on the basic thesis above, we can define the r^{th} neighborhood hash of graph. Given a graph $G =< N, E >$, $N = \{n_1, n_2, ..., n_{|N|}\}$. The neighborhood hash operation produces a set of new bit label S, denoted as $S = \{S^1, ..., S^R\}$, each iteration is given by:

$$S^r = \{NH^r(n_1), NH^r(n_2), ..., NH^r(n_{|N|})\} \quad (8)$$

where $|N|$ stands for the size of S and R indicates the maximum order of neighbors.

In other words, the neighborhood hash operation of a whole graph is executed by replacing the label of each semantic node using the neighborhood hash value $NH^r(n)$, where $r$stands for the r^{th} iteration.

C. Semantic Linear-Time Graph Kernel

Suppose there are two graphs denoted as $S1$ and $S2$. The similarity between $S1$ and $S2$ can be efficiently calculated by simply comparing the set of bit labels $S1$ and $S2$. We define the semantic linear-time graph kernel as follows:

$$k(S_1, S_2) = \sum_{r=1}^{R} k^r = \sum_{r=1}^{R} \frac{|S_1^r \cap S_2^r|}{|S_1^r| + |S_2^r| - |S_1^r| \cap |S_2^r|} \quad (9)$$

Where $|S_1^r \cap S_2^r|$ denotes the number of the common labels between two sets of bits labels.

Let $K_r \epsilon R^{|S||S|}$ $(0 \leq r \leq R)$ be the kernel matrix of r^{th} iterations,S denotes the set of graphs and K_{ij} represents the similarity between S_i and S_j. Jaccard-Tanimoto coefficients often are utilized to represent the similarity between two sets of discrete labels. Thus, the similarity between two graphs can be calculated on the basis of this coefficients,$k(\cdot)$.

Our graph kernel is a linear sum of several valid kernels as in Equation 9. It will be positive semi-definite[8], which guarantees the validness of the designed kernel. Bit-independent logical can be generally executed in one clock (a unit time) if the bit length T is no more than the bit size of the processor architecture, which is a critical property for developing linear-time graph kernel. Therefore, the complexity of neighborhood hash operation for all nodes is of the order $O(Nd)$, where N stands for the number of nodes and d stands for the average degree of nodes. Iterative computation time is of the order $O(NdR)$ at most, where R is the maximum iterations, which is much more efficient than the shortest-path graph kernel ($O(N^4)$) or the walk-based graph kernel at least ($O(N^3)$). Calculating kernel matrix helps us classify instances, since similarities of instances are characterized in the matrix. In the subsequent experiments, the proposed graph kernel is combined with Support Virtual Machine (SVM) to perform a multi-class classification task of images.

V. Experiments

In this section, we validate the efficiency and precision of the proposed method by performing an image classification. Experiments have been carried out on Core114 [9] and Coil100 [10] datasets. The comparison is made with hand-labeled classification and using SVMs for classification [9].

A. Experimental setting

Because semantic information depicts the affinity between neighbors, the semantic information between neighbors can be represented by strong and weak. If two neighbors have much more common boundaries than others, the semantic information between these two neighbors can be regarded as strong; otherwise, it is regarded as weak. Based on this, the semantic relationship between two neighbors is considered as follows:

$$R(i,j) = \begin{cases} strong, & \text{if } C_{ij}/(B_i + B_j) \geq T, \quad (10) \\ weak, & \text{otherwise} \quad (11) \end{cases}$$

Where B_i and B_j denote the boundary of region i and j respectively, C_{ij} represents the common boundary between region i and j, and $R(i,j)$ denotes relationship between two neighbors. T is a threshold and T is set as 0.15 in our experiment.

B. Results

In this section we show the results of classification in two aspects: error rates and time complexity. We compare the precision of our approach with that of $SVMs$ based models. Table II shows the error rates of SVM based on $KNN\ L_2$,SVM based on $KNN\chi^2$ and our approach, respectively. The results show that the error rate of our approach is lower than those of other two methods.

TABLE II
Error rates with KNN and Our method

	$KNN\ L_2$	$KNN\chi^2$	Our approach
$Corel14$	47.7	26.5	19.6
$Corel100$	51.4	35.4	20.4

According to the theory of computer composition principle, all the bit operations are atom computation, which means each bit operation can be finished in one clock. Because all the operations of graph kernel are on the foundation of bit operations, the classification can be finished in linear-time.

VI. Conclusion

This paper extends the traditional graph kernel approach to complex semantic-rich graphs. We design a semantic linear-time graph kernel for image classification and utilize this graph kernel to calculate the similarity between graphs. We overcome the deficiency of traditional single tag, make the attributes of nodes as multiple labels and fix the dimension of the label space. The similarity between graphs can be efficiently computed, compared to the SVMs based kernels. Evidently, the proposed method is effective to calculate the similarity between semantic-rich graphs.

Acknowledgment

This work is supported by National Key Technology Research Development Program of China (NO.2013BAD15B06).

References

[1] A. I. H Kashima, K Tsuda, "Kernels for graphs," *In Kernel Methods in Computational Biology*, 2004.

[2] F. B. Zaid Harchaoui, "Image classification with segmentation graph kernels," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1–8.

[3] Q. H. Z. H. Li Peng, Zhiying Zhang and H. Zhuge, "Designing a novel linear-time graph kernel for semantic link network," *CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE*, 2015.

[4] E. J. V. L. K. M. K. M. B. F. B. Nino Shervashidze, Pascal Schweitzer, "Weisfeiler-lehman graph kernels," *Journal of Machine Learning Research*, vol. 12, no. 3, pp. 2539–2561, 2010.

[5] B. A. I. Ben Salah M, Mitiche A, "Multiregion image segmentation by parametric kernel graph cuts," *Image Processing, IEEE Transactions on*, vol. 20, no. 2, pp. 545 – 557, 2011.

[6] A. I. Salah MB, Mitiche A, "Effective level set image segmentation with a kernel induced data term," *Image Processing, IEEE Transactions on*, vol. 19, no. 1, pp. 220 – 232, 2010.

[7] V. O. Z. R. Boykov, Y., "Fast approximate energy minimization via graph cuts," in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, 1999, pp. 377 – 384.

[8] O. B. O. B. Matthias Hein, Matthias Hein, "Hilbertian metrics and positive definite kernels on probability measures," *Proceedings of AISTATS 2005*, pp. 136–143, 2005.

[9] H. P. V. V. Chapelle, O., "Support vector machines for histogram-based image classification," *Neural Networks, IEEE Transactions on*, vol. 10, no. 5, pp. 1055 – 1064, 1999.

[10] N. S. Nene SA, "Murase h. columbia object image library (coil100," *Columbia University*, 1996.