Image Classification Using The Web Graph

Dhruv Mahajan
Yahoo! Research
Bangalore
India
dkm@yahoo-inc.com

Malcolm Slaney
Yahoo! Research
Santa Clara, CA
USA
malcolm@ieee.org

ABSTRACT

Image classification is a well-studied and hard problem in computer vision. We extend a proven solution for classifying web spam to handle images. We exploit the link structure of the web graph: a web page related to a given category is normally linked to other pages describing related objects. Our approach combines information from the web-graph structure with semi-supervised learning from all the unlabeled images to create a superior image-classification model for multimedia data. We show that fusing image, text and web-graph features gives a 12% improvement (in the area under the ROC curve) over content features alone in an adult image-classification experiment.

Categories and Subject Descriptors

I.5.2 [**Pattern Recognition**]: Design Methodology-Classifier design and evaluation

General Terms

Algorithms

Keywords

Algorithm, image classification, web graph

1. INTRODUCTION

Motivation: Multimedia recognition is a hard problem. Much effort has been directed towards recognizing objects and classifying images (and other multimedia data) using only the pixels in the image. In this paper we demonstrate that including the context in which an image appears is an important feature in an image-classification experiment.

We describe a framework for multimedia classification that exploits the fact that the *context* of an image provides useful information for image classification. A cricket website is likely to point to other websites with cricket-related information and images. The latest photograph from a photographer that uploads hundreds of hockey pictures is likely to be another hockey picture.

Our solution optimizes a single function of the image features, the text around the image, and the web graph. We learn a linear classifier on the image and text features. We exploit the web graph, or hyperlink information, by doing graph regularization to constrain the predicted scores to vary smoothly between the linked pages. We extend a web-page-spam detection approach [1] to predict the image score in a single optimization framework.

We test our algorithm on an adult- or offensive-content-recognition task. While we demonstrate our approach using an image-classification problem, we believe that the same formalism applies to other recognition tasks on the web, and to other types of content networks like Flicker and Facebook.

Our work makes two contributions. We show 12% better performance (as measured by the area under the ROC curve) when we add the web graph to content features. We further show that a form of semi-supervised learning allows us to benefit from unlabeled images, and thus gain two percentage points of accuracy.

Related Work: There is much research related to the work described here. Yet none of this research spans the important dimensions we consider here.

Image classification is a well-known problem, with work spread over many decades. The common works explore different kinds of features, and different kinds of machine-learning approaches to learn from the data and better make a decision. Notably, several groups have tackled the Caltech256 problem, a task where a system must automatically classify a large set of canonical images into one of 256 different classes [7].

Several groups have noted that web pages are likely to connect to other web pages on the same topic [4]. We also benefit from the (local) coherence of the web. The web-spam paper by Abernethy [1] gives a good review of algorithms that incorporate web-graph information in their decisions. Alas, none of their references applies to images.

Adult-image detection has been studied as a particularly interesting type of image-classification problem. Starting with the original work by Forsyth that uses color and texture features [6] we now have systems that use motion (for video) [8], but they do not exploit context.

We do not know of any work that uses the web graph to improve image-classification results. One work [3] creates a low-dimensional embedding from the web graph for clustering images. In this work we show how to unite the web graph and image features to build a better image classifier.

MM'10, October 25–29, 2010, Firenze, Italy.

2. APPROACH

We extend an approach first popularized in the web-spam detection domain [1] to the images linked to web pages. For each image we calculate pixel-based and text-based features (which are concatenated into a vector $\mathbf{x_i}$) and take into account an image's position in the web graph (based on the directed edges E).

2.1 Basic Algorithm

The basic classification algorithm, in this work, is a simple large-margin linear classifier. There are N web pages. We have $l < N$ labeled nodes and each label is denoted as y_i with a value of either +1 or -1. We want to find a classifier $\mathbf{w}$ that predicts the correct label for each (labeled point) and minimizes the loss of the function

$$\Omega_{\mathbf{w}}(\mathbf{w}) = 1/l \sum_{i=1}^{l} R(\mathbf{w} \cdot \mathbf{x_i}, y_i) + \lambda_1 \mathbf{w} \cdot \mathbf{w}, \quad (1)$$

where λ_1 is a regularization parameter that sets the importance of using a small $\mathbf{w}$ and limits the complexity of the decision. The function $R(\cdot)$ controls the effect of errors and in this work is set to $R(s, y) = [1 - s * y]_+^2$, where the $(x)_+$ operator is equal to $\max(0, x)$.

This simple model is enhanced by learning an additional slack variable per node, z_i. The score function (s_i) for a page becomes $s_i = \mathbf{w} \cdot \mathbf{x_i} + z_i$ and we minimize

$$\Omega_s(\mathbf{w}, \mathbf{z}) = 1/l \sum_{i=1}^{l} R(s_i, y_i) + \lambda_1 \mathbf{w} \cdot \mathbf{w} + \lambda_2 \mathbf{z} \cdot \mathbf{z}, \quad (2)$$

where $\mathbf{z}$ combines the elements of z_i in to one vector.

We expect neighboring nodes, as determined by the web graph, to have similar scores. Hence, we add a regularization

$$\Omega_{\mathbf{w}}(\mathbf{w}, \mathbf{z}) = \gamma \sum_{(i,j) \in E} a_{i,j} \Phi(s_i, s_j), \quad (3)$$

where $(i, j) \in E$ describes all web nodes that are connected, γ is another regularization parameter and $a_{i,j}$ is the number of links from node i to node j. Φ is a function that compares the scores from two different nodes. There are two kinds of information flows, symmetric and asymmetric. A simple choice for Φ is a squared loss term $\Phi_s(f_i, f_j) = (f_i - f_j)^2$. An asymmetric version looks like $\Phi_a(f_i, f_j) = \max(0, f_j - f_i)^2$. We blend these two conditions with

$$\Phi(f_i, f_j) = \alpha(f_i - f_j)^2 + (1 - \alpha)\max(0, f_j - f_i)^2. \quad (4)$$

The total cost function is

$$\Omega(\mathbf{w}, \mathbf{z}) = \Omega_s(\mathbf{w}, \mathbf{z}) + \Omega_{\mathbf{w}}(\mathbf{w}, \mathbf{z}). \quad (5)$$

In this formula, we want to find the values for a discrimination vector $\mathbf{w}$ and slack variables z_i that minimize the sum of these two different terms: one that says the labeled nodes are predicted correctly, and *all* connected nodes, whether they are labeled or not, have a similar score. This second term is what allows the system to learn from the web graph and reflects a form of semi-supervised learning.

2.2 Web-Image Classification

The spam-classification algorithm we described above has a single feature vector $\mathbf{x_i}$ associated with each web page. However, images (and other multimedia objects) are treated differently on the web. Each image has a separate feature vector where as the web page does not have any features associated with it. In this section, we propose modifications to the classification algorithm that allows us to handle web pages and their images in the same framework. This is one of the main contributions of our work.

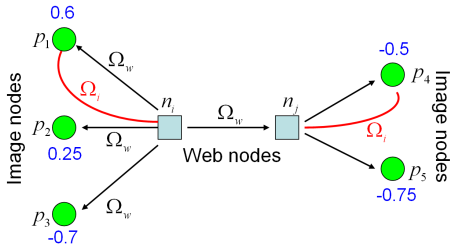

Figure 1: Graph for web-image classification.

In order to handle multiple images per web page, we add image nodes to the web graph as shown Figure 1. We add directional links/edges from the web page to its corresponding multimedia objects or images (black edges). These edges have the same asymmetric properties as the edges in the web graph. Hence, the edge set E in Equation 3 has now *web node* $\rightarrow$ *image node* edges.

Web pages and images nodes have different information and constraints. We note this missing information and add a new constraint between a web page and its images. Web pages contain no feature data—$\mathbf{x_i}$ does not exist for these nodes—and are always unlabeled. Let W denote the set of all web nodes (corresponding to web pages). Then the (classification) score is

$$s_i = z_i, \ i \in W. \quad (6)$$

Image nodes, on the other hand, have both features $\mathbf{x_i}$ and slack variables z_i.

In order to propagate image information to its web page, we assume that a web page is relevant to the classification task (label = +1) at hand if even one of its images is relevant. Hence, a web page takes a score that is the maximum of its image scores. This is done with an additional regularization term (red edges in Figure 1) . We encourage consistency by adding this new term to the loss function

$$\Omega_i(\mathbf{w}, \mathbf{z}) = \gamma_2 \sum_{i \in W} \Phi_s(s_i, \max_{\substack{(i,j) \in E \\ j \notin W}} s_j). \quad (7)$$

We sum over all web pages (described by the set W) and $\Phi_s(\cdot)$ describes the consistency between the score for web page i and the maximum of its image scores. Further, since there are no features associated with the web pages (Eqn. 6), the corresponding slack variables are not balanced against a decision based on the feature data. Hence there is no reason to regularize or limit them in Ω_s. Equation 2 becomes

$$\Omega_s(\mathbf{w}, \mathbf{z}) = 1/l \sum_{i=1}^{l} R(s_i, y_i) + \lambda_1 w \cdot w + \lambda_2 \sum_{i \notin W} z_i \cdot z_i. \quad (8)$$

Note the third term sums over the slack variables for images only. The complete loss function is (Eqns. 3, 7, and 8)

$$\Omega(\mathbf{w}, \mathbf{z}) = \Omega_s(\mathbf{w}, \mathbf{z}) + \Omega_{\mathbf{w}}(\mathbf{w}, \mathbf{z}) + \Omega_i(\mathbf{w}, \mathbf{z}). \quad (9)$$

To compute the scores, we jointly optimize $\mathbf{w}$ and $\mathbf{z}$ using Newton's method as detailed in Section 3.2.

2.3 Features

We use a combination of text and image features to describe each image in the web graph. For text features, we use a search engine to gather the words around each image, its URL and abstract. We then compute a low-dimensional feature vector by summarizing the words, after stemming, in the positive class using a latent-semantic indexing (LSI) model. Thus for each image we form a 100 dimensional word histogram.

Likewise, we use a deep-belief network (DBN) or convolutional neural network (CNN) to represent each image as a point in a low-dimensional space. DBNs are an auto-encoder network that reconstructs with minimum error the original image from the low-dimensional representation. The specifics of our three-layer DBN are described elsewhere [9]. We reduce the dimensionality of 1024-dimensional image features to 500 by doing a PCA transformation.

3. APPLYING TO THE WEB

We apply this image-classification algorithm to a portion of the real web. We use an offensive-content recognition task to explore the behavior of our image-classification algorithm.

3.1 Data

We started with a connected subgraph (of the entire web) that contained a number of images that we had already labeled as offensive or not. This gave us a web graph with a total of 859k nodes. Then, we consolidated the nodes that provided little information in our graph by detecting the web pages that had a single in and out link (and thus no images.) We removed these unitary nodes from the graph and linked the incoming web page to the outgoing web destination. By this process we reduced the graph so it contained 83k web pages and 211k attached images for a total of 295k nodes. Only 1.3% of our images are labeled, with 1291 positive labels and 1405 negative labels.

3.2 Implementation

Given the consolidated web graph, we fetch all images and precalculat all features. We randomly split the 2696 labeled images (1291 positive and 1405 negative) into a training set of size 2000 and a test set of 696 labels. We obtain smaller training sets by sub-sampling the training set. However, the testing is always done with the same test set of 696 images. In order to reduce the result variance for smaller training sets, we repeat the experiment with 7 different training sub-samples of that size and then take the median.

Our objective function is convex. Because of the *max* function (Eqn. 7), our cost function is no longer differentiable at a small number of points. However, the contribution of this term is small (Sec. 4.1). In general, we did not find this to be an issue in our optimization. We minimize the function by the iterative Newton method [5]. Each Newton step requires computation of the inverse Hessian matrix which is very large in our case. The Newton step is instead computed by solving a system of linear equation using linear conjugate gradient [5]. The method involves computing the Hessian vector product which can be done very fast since our Hessian is sparse.

4. RESULTS

There are two novel aspects to this work. Most importantly our work includes information from the web graph when judging image content (Sec. 4.2). A secondary benefit of our approach is that we can take advantage of unlabeled data. This algorithm demonstrates a form of semi-supervised learning (Sec. 4.3). First we describe how the parameters were tuned.

4.1 Parameter Tuning

Our method requires we determine the values of 6 meta parameters α, λ_1, λ_2, γ and γ_2 and edge weights $a_{i,j}$. For *web node* $\rightarrow$ *web node* edges, $a_{i,j}$ is set to the number of links from i to j. We found that for most of the web nodes, $a_{i,j} = 1$. For *web node* $\rightarrow$ *image node* edges, we found that $a_{i,j} = 2$ works well for all the experiments. For α, we did not see much asymmetric behaviour in the links. Setting α in the range $0.45 - 0.65$ works well for most of the experimental settings. Hence, we fix α to be 0.55. Similarly, for γ_2 in $\Omega_i(\mathbf{w}, \mathbf{z})$(Eqn. 7), setting it in a broad range of $0.2 - 0.35$ times γ works well. We choose $\gamma_2 = .25\gamma$. For choosing the remaining parameters we held out 20% of our training data, for each training-set size and experiment, and used it as a validation set. We used the validation set to find the parameters λ_1, λ_2 and γ over a $5 \times 5 \times 7$ grid that gave us the highest classification score. The results shown in the subsequent sections are after cross validation.

4.2 Benefit of the Web Graph

In this section, we show the performance of our algorithm. In order to highlight the role of different components—image features, text features and web graph, we consider the different variants of our algorithm. The list is not exhaustive but clearly demonstrates the role each component plays.

Image features Only: We set the parameters γ and γ_2 to zero. λ_2 is set to a high value (1000) to force the slack variables towards zero. This has the effect of learning a linear classifier on features with a regularizer.

Web Graph Only: We set the feature vector $\mathbf{x_i}$ at each image node to be zero. Hence, the only contribution comes from the web graph.

Text features plus Web Graph: We set the feature vector $\mathbf{x_i}$ to the 100-dimensional text vector (Section 2.3).

Image Features plus Web Graph: We set the feature vector $\mathbf{x_i}$ to the 500-dimensional image vector (Section 2.3).

Image plus Text features plus Web Graph: We set the feature vector $\mathbf{x_i}$ to a 600-dimensional vector formed by combining the text vector and image vector.

We evaluate our performance by looking at the area (AUC) under the receiver-operating curves (ROC). ROC plots the true-positive rate versus the false-positive rate as we vary a decision threshold.

Figure 2 shows the performance of different versions of our algorithm as a function of the size of the labeled set. Since labeling data is expensive, we are interested in how the algorithm performs as we label more data, and thus increase the size of our training set. In the best case only 1.3% of the images in our database are labeled.

The bottom line of Figure 2 shows our baseline performance: a linear classifier operating on only the image features with no web-link information. Using only the web graph dramatically improves our results over the image features, for all training set sizes. This is surprising given

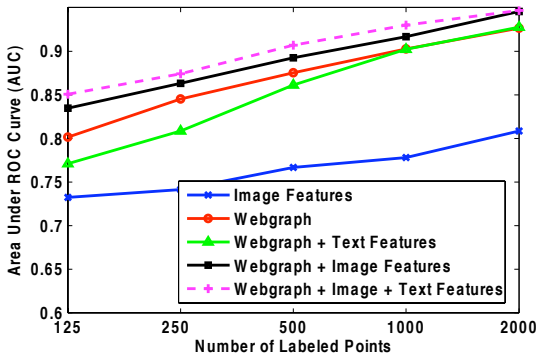

Figure 2: AUC for variations of our algorithm as we vary the training size.

that we are trying to classify the image content. Adding text and image features to the web-graph-based classifier further improve the results. However, adding text features help only for the smaller training sets (top magenta curve vs. black curve). Hence, using all the information available—web graph, text and image features—produces the best results. The ROCs shown in Figure 3 using all our training data further validates our performance.

4.3 Benefit of Semi-Supervised Learning

Table 1 shows the contribution of semi-supervised and web-graph learning in our task. We can add these two aspects one by one in our framework. A conventional image-classification system uses only the labeled images for training data. This amounts to learning a linear classifier on the labeled data, so all decisions on the unlabeled nodes are made using this classifier only (AUC = 0.81). We add the semi-supervised learning (SSL) [2] aspect by constructing a K-nearest neighbors graph for the images in our data using the image features. We then do graph regularization using our framework. However, the performance goes down (AUC = .76). This can be because of two reasons. First, our features may not be strong enough to compute K-nearest neighbors accurately. Second, the total number of images is small and hence their nearest neighbors (based on visual similarity) might not be meaningful.

A more powerful form of semi-supervised learning is available because of the web graph. The edges in the graph are now based on contextual similarity between the linked websites rather than the visual similarities and hence, provide a more robust signal. This gives us the biggest boost

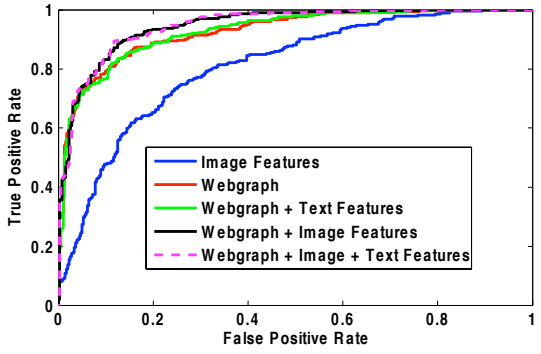

Figure 3: ROC for variations of our method.

Table 1: Overall AUC (higher is better, 1.0 is perfect) showing the effects of semi-supervised learning.

Classifier built using only labeled images	0.8086
Nearest neighbors in feature space	0.755
Labeled images plus web-graph regularization	0.9263
Web graph plus all features and data	0.9465

(AUC = 0.93). Finally, we get the highest performance when we add the text and image features (AUC = 0.95). Hence, both semi-supervised and web-graph aspects of our method are vital for the performance of our algorithm.

5. CONCLUSIONS

We have demonstrated a multimedia-classification system that combines image, text, and web-graph features. All of these features are useful, but, as shown in the web-spam work, the web-graph information, which specifies which images are close to each other on the Internet, is most valuable. We anticipate this web-graph information will benefit other image-classification systems. Since our multimedia objects live in a connected world, it is important to include their context when judging their contents.

6. REFERENCES

[1] J. Abernethy, O. Chapelle, and C. Castillo. Graph regularization methods for web spam detection. *Machine Learning Journal*, 2011, to appear.

[2] Y. Bengio, O. Delalleau, and N. Le Roux. Label propagation and quadratic criterion. In O. Chapelle, B. Schölkopf, and A. Zien, editors, *Semi-Supervised Learning*, pages 193–216. MIT Press, 2006.

[3] D. Cai, X. He, Z. Li, W.-Y. Ma, and J.-R. Wen. Hierarchical clustering of www image search results using visual, textual and link information. In *MULTIMEDIA '04: Proceedings of the 12th Annual ACM International Conference on Multimedia*, pages 952–959, New York, NY, USA, 2004. ACM.

[4] B. D. Davison. Topical locality in the web. In *SIGIR '00: Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 272–279, New York, NY, USA, 2000. ACM.

[5] R. Fletcher. *Practical Methods of Optimization*. John Wiley & Sons, New York, second edition, 1987.

[6] D. A. Forsyth and M. M. Fleck. Automatic detection of human nudes. *Int. J. Comput. Vision*, 32(1):63–77, 1999.

[7] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical Report 7694, California Institute of Technology, 2007.

[8] C. Jansohn, A. Ulges, and T. M. Breuel. Detecting pornographic video content by combining image features with motion information. In *MM '09: Proceedings of the Seventeen ACM International Conference on Multimedia*, pages 601–604, New York, NY, USA, 2009. ACM.

[9] M. Ranzato, Y. Boureau, and Y. LeCun. Sparse feature learning for deep belief networks. In *Advances in Neural Information Processing Systems (NIPS 2007)*, 2007.