Adult Website Classifier

Saikat Sen
CS229 Machine Learning Course Project, Dec 2010

Abstract

The goal of this project was to detect adult websites and pages that are not safe for kids. We use five different techniques. We create an adult vocabulary and use a composite classifier formed of multiple Naïve Bayes classifiers to classify pages based on url, title, keywords and content. We use hue, saturation and histogram of gradients to train random forests, different boosting classifiers and MLP on boxed images for local image classification. The intent is to then use a Viola-Jones or Haar approach to classify images globally. We show an edge-detection technique that works better than Canny's for some images. We show an extension to Markov chains that can help detect edges. The intent is to use classifiers such as SVMs with Gaussian kernels to use edge information in detecting body parts. Lastly, we propose AdultRank, a ranking metric that serves as an indicator of the adultness of a page. All the techniques together can be used effectively to detect adult web sites and pages. The only overlap this work has with previous related work is in image recognition using the features we have used and edge detection techniques.

Introduction

Website classification is an old problem. Internet Explorer labels websites as phishing and malware. Google leaves out malware sites from its search results. The goal of this project was to build a classifier that can classify websites and web pages as adult, i.e. sites and web pages that are unsafe for kids.

Applications of this classification are many. Parents don't want their kids exposed to adult content. Some adults find porn images offensive. Some governments ban porn sites and have an ongoing requirement to detect them. Many porn sites have a malware payload and install rootkits, adware, spyware and other viruses, so guarding against them is an additional safety measure.

In general, sites and pages could be classified as adult sites based on many factors such as adult images, sexual content, violent content, racist sentiments, extreme radical views etc. The scope of this project is limited to the first two categories. We try to classify pages based on metadata and try to find good classifiers for adult images. Video classification was out of scope for this project but can be done by analyzing individual frames. We also extend Markov's model, propose a ranking metric *AdultRank* and propose a new convolution filter for edge detection.

Strategy

We employ three main techniques: image analysis, text analysis and ranking. For text analysis, we inspect page title, keywords, url and content. For image analysis, we use different image recognition techniques. The OpenCV package was used for image recognition and ML classifiers. For ranking, we propose AdultRank, a ranking metric similar to PageRank.

1 Url Text Classification

1.1 Url Features

The following metadata of web pages were used for adult classification:

- Meta tag: "rating". There are some standards that sites can use to indicate adult content but none that we saw use the "rating" meta tag. If the rating tag is found to be *adult* or *restricted*, the page is classified as adult. The code was not included in the final toolset since this analysis can be done independently without using machine learning techniques.
- Title: if the page title contains an adult word, the page is classified as adult.
- Url: if the page title contains an adult word, the page is classified as adult. The chosen implementation is naïve: it looks if the words from an adult dictionary exist in the url as

substrings. A proper implementation would parse the url into words with an optimum match, take the site content into consideration and then do a dictionary lookup. As an example, we consider tit to be an adult word but this false positives sites with “title” in the url. As an example of requiring site content to be considered, “google” can be parsed as the mathematical number google, and “go” “ogle” – the site content would help determine which of the two parsings is more appropriate.

- Keywords: adult sites and pages tend to contain adult words in their list of keywords for better search engine rankings. We search for the keywords in an adult vocabulary.
- Content: Page content is an important determinant. Sites such as TheOnion.com are adult sites with expletive but non-sexual content. Certain blog pages and similar user-contributed content pages (comments on news articles, discussion forums etc.) with expletives cannot be filtered out by any method but content filtering.

1.2 Adult Vocabulary

In the absence of a good online adult dictionary, we constructed one using a custom application and using the Princeton WordNet lexical database. The custom application allows the user to choose a source set of words, outputs the synonyms in each iteration, and allows the user to classify the synonyms as adult, gray and clean before proceeding with the next iteration with the adult synonyms. It is essential to classify the synonyms in each iteration, otherwise the word bag escalate to size in thousands due to words with multiple meanings such as *tool*. We iterated until a new iteration didn’t output any new words. Gray and clean words are not iterated upon. The final list includes two files: adult.txt and gray.txt. Adult.txt contains confirmed adult words that we would like to filter, gray.txt contains words that we would like to filter but can be used in adult and non-adult contexts: the goal is for the classifier to learn the appropriate weights on all words during training.

Our vocabulary contains **106** blacklisted adult words and **26** gray words.

Limitations of the current classification method include not being able to parse different forms (plural, tenses, conjunctional forms) of the parent word – this can be easily solved by preprocessing words before looking up the adult dictionary. Standards preprocessing techniques such as stemming and skipping stop words were not used but ideally should be.

1.3 Url Data

A custom app was written that could start with a starting set of urls and find all the outbound http and https links one level deep. The app allows filtering out urls in each iteration and automatically filters out urls with schemes such as mailto and about. Schemeless urls such foo.html that are clearly internal links were filtered out in each iteration.

Using this method, we collected 450 additional porn urls with a starting set of 11 porn urls and 1500 good urls with a starting set of 20 good urls. Both sets were manually checked to make sure the set contains correct data. Most could be easily discerned from the urls themselves.

A custom app was written to output data files for the good and adult sites in csv format – 8 data files were generated: adult and good for title, keyword, url matching and content.

Caution: many porn sites will install malware (rootkits, adware, spyware, viruses) on your system via drive-by attacks.

1.4 Url Text Classifier

The text classifier classifies sites and pages based on the features mentioned in the preceding “Url Features” section.

The Text Classifier is a hierarchical composite classifier with a Naïve Bayes classifier each for title, url, keywords and content. We experimented with the individual classifiers – constructing the composite classifier can be easily done: if any of the classifiers outputs 1 indicating adult, the final decision is taken as ‘adult’.

1.5 Url Text Classifier Results

The Naïve Bayes classifier works well in almost all cases with high accuracy, high TP rate and a consistently low FP rate. Including more blacklisted words into the vocabulary and reducing the number of gray words improved the metrics, and a bigger training set and vocabulary can further improve the performance of these classifiers. In the image below, content is shown to have quite a few false positives – this was with 106 blacklisted adult words and 26 gray words.

Algo		FP rate	TP rate	Acc
		0	0	
Naïve Bayes	Title	0	86.7679	96.88934
	Keyword	0.133333	88.93709	97.2973
	Url	0.333333	95.44469	98.67415
	Content	1.333333	65.07592	90.77002

2 Image Classification

The goal of the image classification is to find adult images. We tried both global classification and local classification with boxed images. In this form, the strategy is to train different classifiers to male and female organs from different angles. We did the project with just one class of adult images due to shortage of time but the intent is to extend the approach and train other classifiers similarly.

In a complete classification model, we imagine training different classifiers for different classes of adult images and compositing them. The final classification would use a weighted sum over the learned individual classifiers as is done in Boosting models:

$$H(x) = \text{sign}\left(\sum \alpha_t * h_t(x)\right)$$

where the summation is done over t = 1 to T = #classifiers and the α_t parameters are learned using the boosting algorithm.

2.1 Image Data

Craigslist personals and other sections were used as the data source. A custom app was written to look for images in the personals section, enumerate the html img tags, crawl those urls and download images from the urls to a local folder. A second custom labeler app was written that allowed easy enumeration through the downloaded images and labeling with outputs organized by labels. Office Picture Manager was subsequently used to box the images for optimal training.

Images were grouped into two positive classes of ~200 images each – where one class corresponds to the frontal and the second class corresponds to profile view. Positive (adult) images were collected from the Personals sections of three different cities. 4000 negative (clean) images were collected from the baby and pets section of two different cities. Because of Craigslist user moderation, these two sections did not contain any positive images.

Note that all boxed images were quality images. Images that were poor resolution, incomplete, hazy, mixed with other body parts or clothes, grayscale and cartoonish were not used for training. We did not employ background subtraction or other image pre-processing techniques.

2.2 Image Features

We mostly tried object recognition using the following features, but also tried edge detection. Time prevented us from training classifiers with edge detection techniques. Two new edge detection techniques were attempted as described below.

Image features used include:

- Color histograms – HSV histograms of images were obtained and hue and saturation histograms were extracted. We tried with 10, 20 and 30 bins for hue and 32 bins for saturation. We experimented with hue separately as a feature and with {hue, saturation} {10/20/30 x 32} bins. As an example, 30 hue bins x 32 saturation bins produced 960 features per image. Hue is a good indicator of skin.
- Histogram of gradients – the gradient at each pixel of an image was calculated as arctan(dy/dx) and mapped to one of 16 buckets in the (-π, π] range; finally the percentage of pixels in each of the 16 buckets was output as an image feature. This produced 16 features per image.

The intent is to be able to run the **Viola-Jones/Haar classifier** that uses a cascaded boosted rejection model to detect body parts in images using the trained local classifiers. We trained the OpenCV Haar classifier successfully using the image database but the 64-bit OpenCV Haar classifier consistently failed to malloc at runtime – we will continue working on this.

2.3 Image Classifiers

We tested the following classifiers for classifying images:

- Random Forests, with 100 trees, max depth of 10, min sample count of 10, regression accuracy of 0, no surrogates, no priors, 15 max categories, forest accuracy of 0.01.
- Boosting classifiers with max depth of 5, 100 weak classifiers, weight trim rate of 0.95 (samples with summary weight < 5% do not participate in the next iteration of training), no surrogates, no priors.
    - AdaBoost – Real, Gentle and Discrete
    - LogitBoost

Splitting criteria used: Gini for real, MiscClass for discrete and least-squares error for LogitBoost and gentle AdaBoost.

2.4 Image Classification Results

We experimented with different features and the two classes of images to see which classifiers fared better, to check the relative importance of features and to determine the optimal values of the different parameters. We tested one run with MLP as well – that took 1.5 hours to run. For the boosting classifiers half the set was presented as training set and half as test set. For random forests, 80% of the data was used as training data. In all cases, both training and test sets had a mixture of positive and negative data. We considered k-fold cross-validation but didn't use it.

2.4.1 Results for Hue Histograms

As the ROC curves show the FP rate in all cases is extremely low (< 1%), with TP rates ~70% for Hue for local. We did not get good results with our set for global detection. Accuracy is high in all cases. This validates that hue is a good indicator of skin and can be used reliably for skin detection in local analysis. All boosting classifiers fared well.

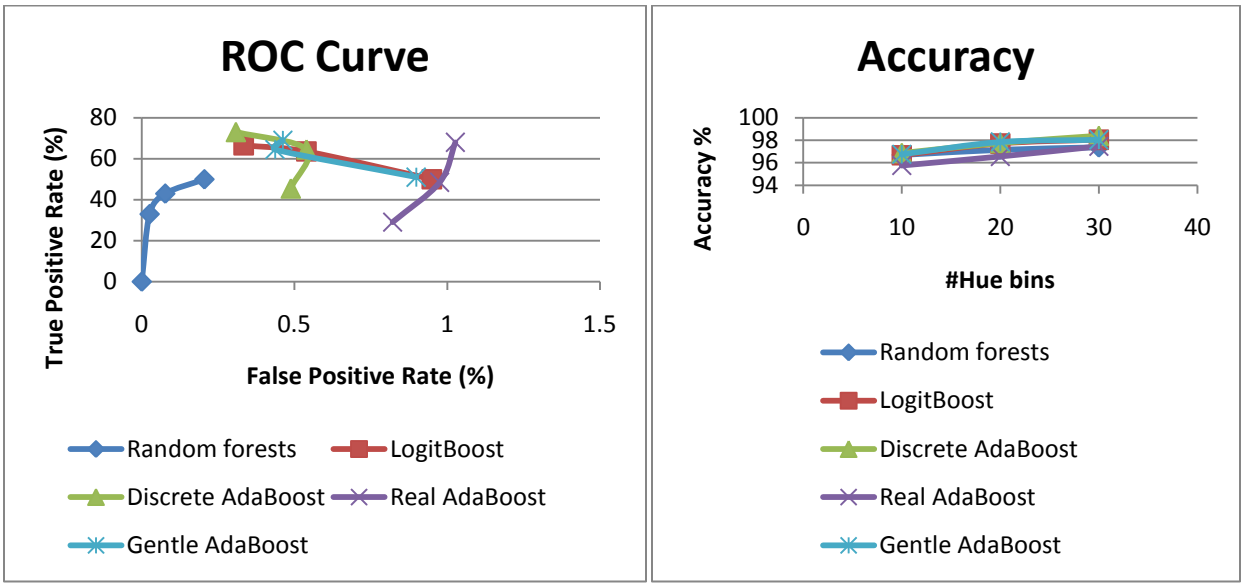


Figure 1. ROC Curves and Accuracy Graph for Feature: Hue, Detection: Local, Image Class: 1.

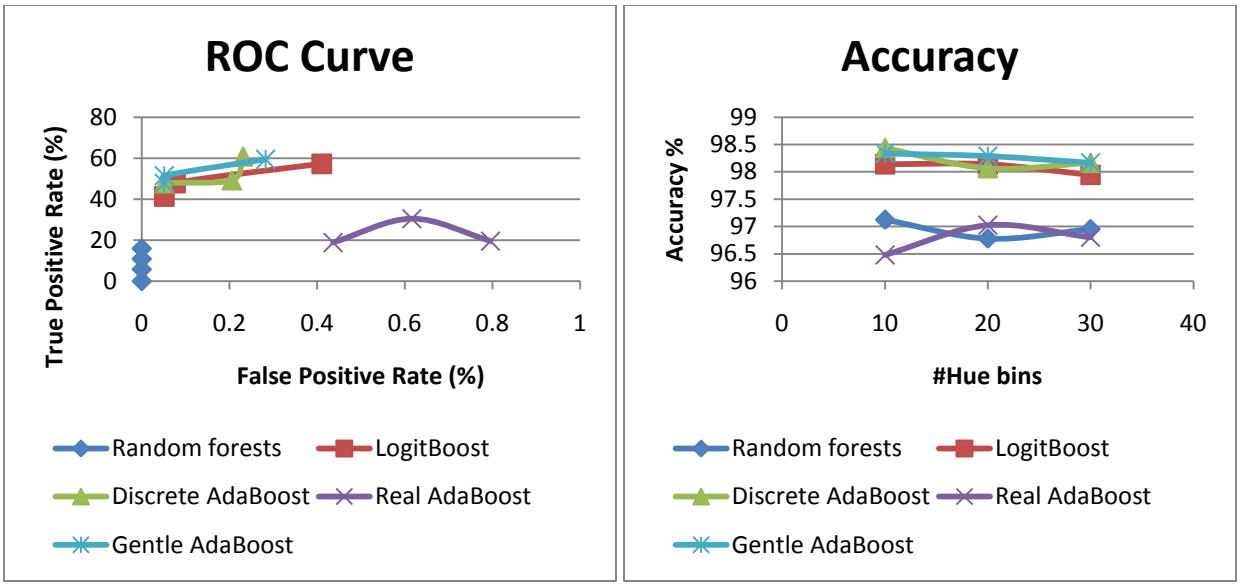


Figure 2. ROC Curves and Accuracy Graph for Feature: Hue, Detection: Global, Image Class: 1.

2.4.2 Results for Hue+Saturation Histograms

As the ROC curves show the FP rate in all cases is extremely low (< 2%), with TP rates with peak ~70% for Hue+Saturation. Results are worse for global analysis. Best models: Gentle and Discrete AdaBoost.

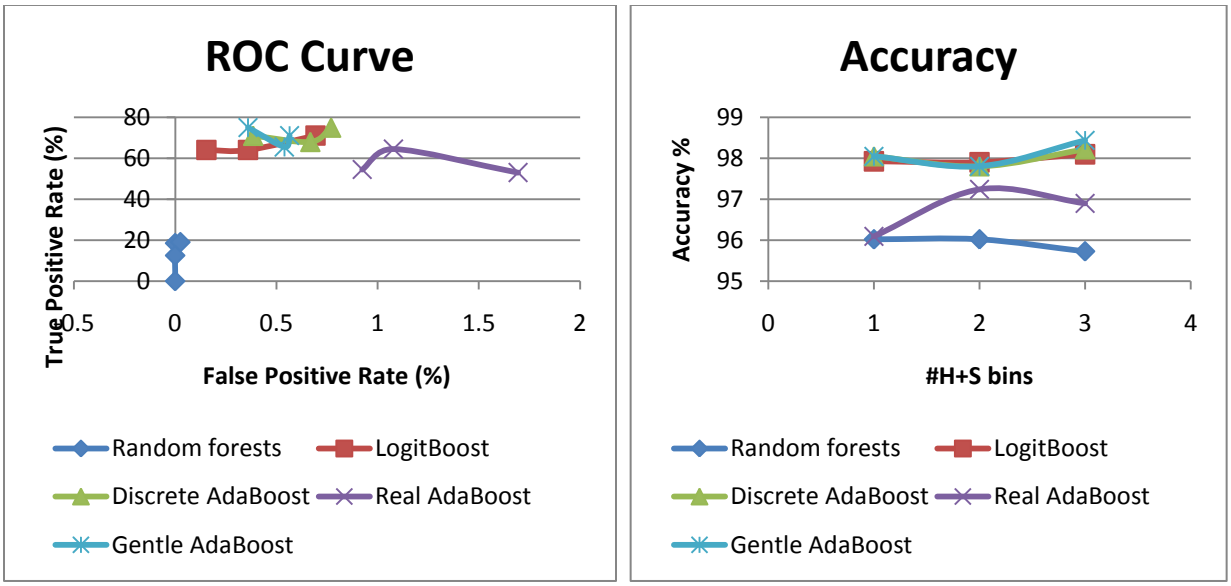


Figure 3. ROC Curves and Accuracy Graph for Feature: Hue+Saturation, Detection: Local, Image Class: 1.

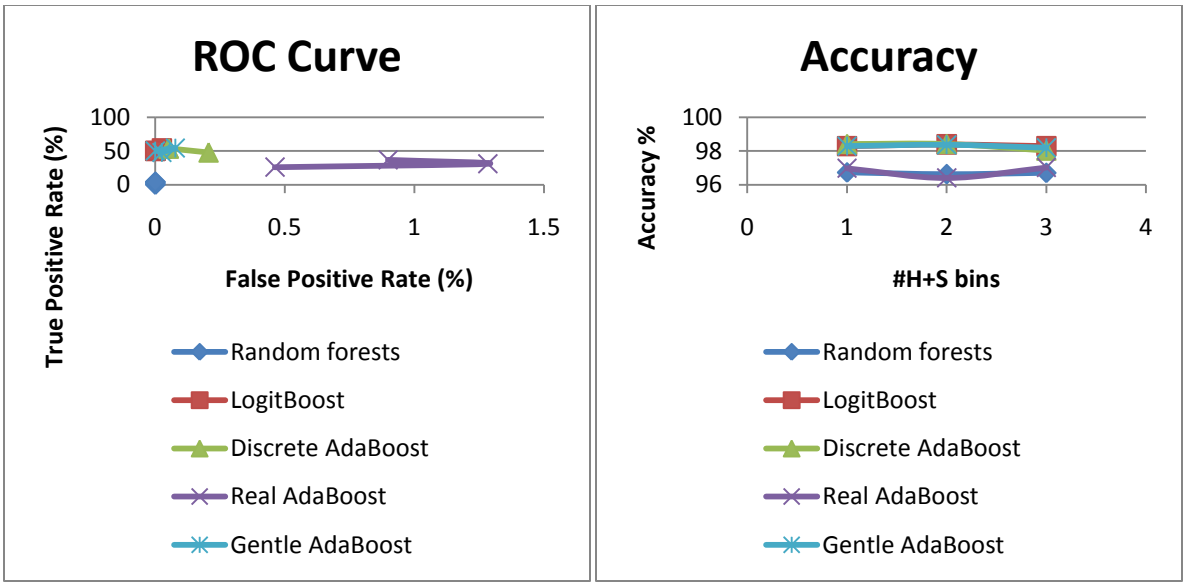


Figure 4. ROC Curves and Accuracy Graph for Feature: Hue+Saturation, Detection: Global, Image Class: 1.

2.4.2 Results for Histogram of Gradients

We calculated the Histogram of Gradients of images – for both full and local classification. With 16 histogram bins, we got the best results (TP rate 65%, FP rate 0.5%) with Discrete and Gentle AdaBoost – the other boosting models and random forests did not fare well. This is kind of expected; full classification is a far harder problem than local classification and we wonder how the histogram of gradients would fare as a feature in global image classification with large data sets.

Results are far better for local classification with Discrete and Gentle AdaBoost returning 80%+ TP rates, < 1% FP rates and ~98% accuracy as the chart below shows.

Algo	TP	FP	TN	FN	FP rate	TP rate
					0	0
R-trees	188	12	3884	196	0.308008	48.95833
LogitBoost	310	44	3852	74	1.129363	80.72917
Discrete AdaBoost	316	26	3870	68	0.667351	82.29167
Real AdaBoost	242	61	3835	142	1.565708	63.02083
Gentle AdaBoost	318	28	3868	66	0.718686	82.8125

Figure 5. Results for Feature: Histogram of Gradients, Detection: Local, Image Class: 1.

2.5 New Filter: DeBrushing

During experimentation we realized that edge detection would benefit from removing the internal pixels of a solid object. We applied a method we call DeBrushing to achieve this. Below we show the results of applying this method to some non-sexual images. The results are promising and seem to be more effective in edge detection for some images as we show below.

The pseudo-code for this method is as follows:

Define a threshold T to be in the range [0, 180]

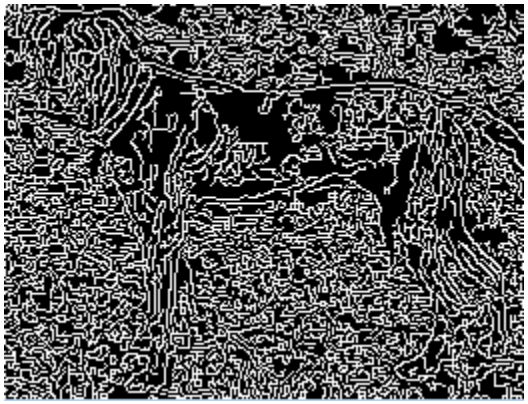
For each pixel P do
Begin
h1 = hue value of this pixel

For each pixel Q that is in the {North, East, NE, SE} of this pixel, h2 = hue value of Q

If (|h1 – h2| < T) for each combination of {P,Q}, color P as red else color P as white.
End

This algorithm does not work well with RGB channels as might be expected. In essence, the above algorithm removes all pixels that have very similar hue values to its neighboring pixels. As an example, following is the result of applying this filter to the image of a dog against a grassy background, with a configurable hue threshold of 8 (two neighboring pixels are considered to be the same hue if their hue values differ by 7 or less). This filter removes all the grass instead of detecting the edges in them.


Below is what Canny's Edge Detector outputs for the same image – it shows the edges in the grass as well. The DeBrush filter performs better in this case.



Below are Mona Lisa's pictures convolved using this filter.

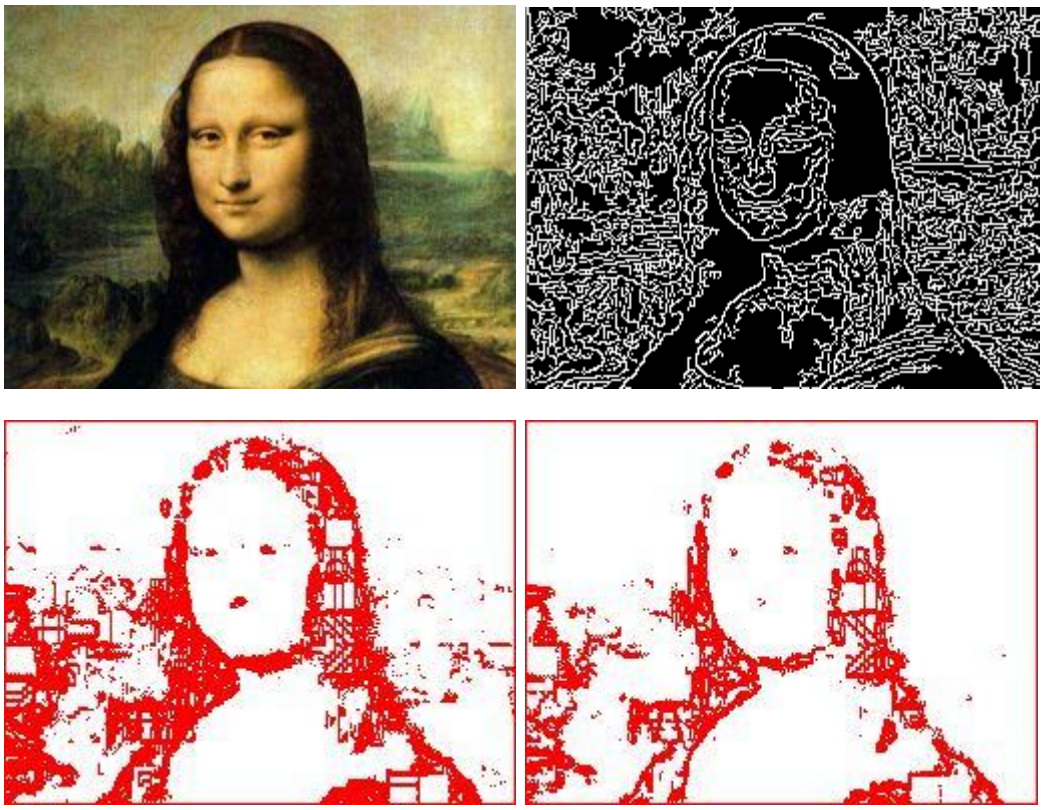


Figure 6. From left to right: Mona Lisa (original), using Canny's edge detector, using my DeBrush operator with a hue threshold of 8, using my DeBrush operator with a hue threshold of 15.

We did not get time to use the convolved images using suitable classifiers for edge detection and using edges as image features but we will continue working on this.

3 Extension of Markov Model

We propose an extension of the Markov Model to detect edges in images. Markov models have not been used in image recognition; this might be relatively slow to run but nevertheless is an interesting technique.

***Observation 1*:** The starting point of a random Markov walk does not matter. Irrespective of the starting point, one finally arrives at the optimal value functions and optimal policy.

***Observation 2*:** At any time, it is possible to deploy multiple agents simultaneously walking the set of states and bookkeeping rewards and transitions with one agent doing periodic policy and value function updates; one would still finally converge and would converge at the optimal value functions and optimal policy.

We do not prove the second observation but it is easy to see why this would hold true.

These two observations imply that Markov chains can be executed on parallel threads in a large state space. This is important, because we apply the Markov model in edge recognition in images with a large state space: the cardinality of the state set is the number of pixels in the image. Before applying to this problem, the distributed walk was implemented on the inverted pendulum problem and worked well. We make one important observation: in a multi-agent Markov walk where there is one thread that does updates to the value functions periodically, the update thread works slow because it performs a lot more computations relative to the other say 100 worker threads. The update thread needs to be prioritized with higher priority and the others as low priority for this model to work well. We added synchronization so the workers would wait for the update thread to complete before proceeding with the next iteration. Without this, the update thread will starve and not get enough time slices to complete one update iteration.

Simplification 1

Consider the case where the result of a transition from a state s upon action a is well-defined instead of a probability distribution, i.e. T(s,a) = s' is well-known instead of being a probability distribution.

In this case, Bellman's equation for the optimal value function:

$$V^*(s) = R(s) + \gamma \sum_{s'} \max_{a \in A} P_{sa}(s')\, V^*(s')$$

Simplifies to:

$$V^*(s) = R(s) + \gamma \sum_{s' \in N(s)} \max_{a \in A} V^*(s')$$

Where N(s) is the set of neighbors of s – the set of states one can transition to from s, i.e.:

N(s) = {s' | s'= T(s, a)} where the transition function T(s, a) gives the state reached from s on taking action a.

This simplification comes handy in case of large state spaces by reducing the $O(N^2)$ term $P_{sa}(s')$ to an O(N) term (N being the cardinality of the set of states), thus reducing computation and memory requirements. In our case, we used a state space proportional to the size of an image in pixels; the runtime memory requirement of our application was in the order of gigabytes before this simplification.

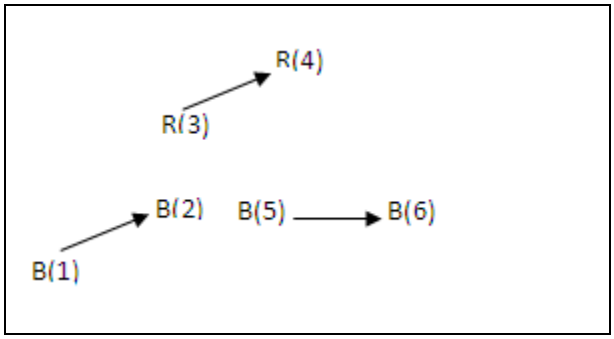
Extension

We build upon simplification 1 in this extension. Consider a case where different transitions are possible from a state s but the transition you want to make does not depend only on the value of the new state but also on a function ψ that is indicative of the similarity between s and s' and the "reward" the transition entails.

Consider the example we use this model in. We want to detect edges in an image. Given any pixel that is on an edge, to draw the edge you would want to transition to the next pixel that has a hue value close to this pixel's hue value. However, there could be multiple candidates among the 4 pixels in the N, E, NE and SE directions to this pixel, and a jump to the pixel closest in hue may eventually lead you to a dead end (as in, going greedy short-term may not be the best choice long-term). The best candidate pixel Q for a given pixel P is the one that is close in hue to P but also neighbors pixels that are close in hue to itself.

Equivalently, we want to transition from P to a pixel Q with high value V but also such that P and Q have similar hues.

For further illustration, consider the following case: B and R are 2 hue values that are completely different. 1, 2, 3, 4, 5, 6 are six pixels laid out with hue labels thus:



1, 2 are neighbors, so are (3,4), (5,6), (2,3) and (2,5). 3 and 5 clearly both have high values. You are trying to decide the best transition from 2. You have a choice to transition from 2 to 3 or 2 to 5. Clearly, to draw an edge the optimal transition is from 2 to 5 because they have the same hue. There are 4 actions for each pixel: transition to the pixel in the direction N, E, NE and SE. In a variant of this example, consider the case where 5 has a lower value V than 3 – the best transition is still 2 to 5, for drawing the edge.

This example illustrates the role of the similarity function. We define the similarity function for this example as:

Ψ(s, s') = 1 – 1/180*Δ where

s, s' are two neighboring states/ pixels and

Δ = | Hue(s) – Hue(s') | is the difference in hues of the two states

The above function is defined such that Ψ takes the value 1 for two pixels with the same hue and 0 for two pixels with hue values differing by 180 and is a real value in [0,1].

Bellman's Equation in this model is:

$$V^*(s) = R(s) + \gamma \sum_{s' \in N(s)} \max_{a \in A} V^*(s') * \Psi(s, s')$$

Where Ψ(s, s') is a real in [0,1]

And the Value Iteration algorithm becomes:

1. For each state s, initialize V(s) := 0
2. For every state, update
   $V(s) := R(s) + \gamma \sum_{s' \in N(s)} \max_{a \in A} V^*(s') * \Psi(s, s')$

We do not include a formal proof but the above algorithm can be easily proven to converge since we replace the probability function by a similarity function with range [0,1]. Our test runs indicate that it converges, although it can take time for a big image.

Putting it all together

The extension and observations above allow us to treat edge detection as a Markov chain problem and detect edges in images fast with multiple threads. The update thread must be high priority and the others lower priority. It helps to synchronize between the threads so while the update thread runs, the others wait.

In the end, one needs to use the DeBrushing method when the run is complete to zero out the internal pixels of a solid object. The benefit of using this is we get more continuous edges than DeBrushing alone.

4 Using edge information in image classification

We did not have time to use edge information in doing classifications. SVMs with Gaussian kernels have been used in the past with reported success even with fuzzy images for object recognition and we would like to try this technique. Histograms obtained by using the Laplace, Canny, Scharr filters to images can be used, so can the DeBrushing filter we propose and the Markov walk.

5 AdultRank

We propose AdultRank, a rank for web pages to indicate a measure of adultness of the page. The concept is somewhat similar to Google's PageRank. Prior to establishing the concept, we observe:

- Porn sites have strong inter-linkage and almost form a closed set.
- Many porn sites link to Facebook and Twitter, both clean sites.
- Some clean sites with unmoderated user-contributed content such as some wiki and blog pages link to porn sites.
- Most other clean pages and sites do not have outbound links to adult sites.
- Craigslist has a personals section which requires age verification and is an adult section but the rest of craigslist is kid-friendly. Some of the Craigslist discussions have expletives.

With this in mind, we propose AdultRank. The range of AdultRank could be anything, but we assume here it to be [0,10].

Let α' be the "intrinsic AdultRank" of a page P. This is a score computed by classifiers we use to classify the page. We do not propose a detailed way to calculate this score in this paper but assume it is calculated using a reasonable heuristic.

Let also α_i be the AdultRanks of the web pages P links to. Let α be the effective AdultRank of P. α is calculated as follows, with w being a real in the [0,1] range (a good estimate is 0.5):

If $\alpha' > \max_i \{ \alpha_i \}$
then $\alpha = \alpha'$
else $\alpha = w * \alpha' + (1 - w) * \max \{ \alpha_i \}$

We do not include a formal proof of **convergence** for this algorithm but it can be shown to converge. This also takes into account the preceding bullet points in this section, clean sites with user content are not unduly marked as adult but they do accumulate rank, the rating of an adult page is not diluted by having outbound links to clean pages, Craigslist personal section and adult discussions are marked as adult but not other sections, AdultRank "flows" between interlinked adult sites and adult sites do not pass on their ranks to outbound links. Considering a threshold (say 7 or 8), a browser could decide to block a site as adult if its AdultRank exceeds the threshold.

We have not yet experimented with the urls database to check how AdultRank fares in practice.

6 Limitations

Local image classification is more accurate than global classification but there are limitations of this method. Certain pictures can be easily classified as adult by a

human judge even if they do not show explicit body parts. This method will also not be effective in judging fuzzy, incomplete, cartoonish and unclear images.

Markov walks are expensive and, unless this method is further optimized, may not fare well on client machines if real-time image classification is the need.

7 Conclusions

We got strong results with metadata analysis, which proves that a Naïve Bayes classifier would fare extremely well for classifying sites and pages based on url, title, keywords and content.

Our experiments on image analysis with even a limited data set shows that global analysis purely based on skin is unreliable but a similar data set with boxed images grouped into classes works reliably with 70% detection rates based on hue alone and 30 hue bins. Saturation seems to be somewhat of a randomizing feature, as whilst we noted monotonically better recognition rates with increasing hue bins, we noticed worse performance when saturation was added to the feature set and did not notice any particular pattern with increasing number of bins as one might expect. Gradient of histograms is a strong feature – we got TP rates exceeding 80%. Just like for face detection, the local analysis method can be used in conjunction with a Viola-Jones classifier to do body part detection in a bigger picture. The downside of this approach is if the analysis needs to be done on a client with low computing resources, this method would be expensive.

The Boosting classifiers (LogitBoost and variations of AdaBoost) outperformed random forests by significant margins in general. Gentle AdaBoost worked very well in most situations. We used the MLP perceptron as well but that takes several hours for a run while the others take a minute or less to train and test. We hit peak TP rates of 73% with Discrete Adaboost in the hue-based local case and 75% with Gentle Adaboost in the hue+saturation-based local case.

For global detection, the boosting classifiers hit peak 50% TP rates but it is hard to tell if this is reliable. At best, we conclude that boosting classifiers can be used as weak classifiers for global analysis.

In all local and global analyses, we noticed low FP rates and high accuracy, part of which is due to the negative data set size being significantly larger than the positive set. AdultRank has a lot of promise and we will experiment with this. We also want to use some of the edge detection methods (such as SVMs with Gaussian kernels) used in the past using outputs from the Laplacian, Scharr Filter, Canny's filter, the DeBrushing method and the DeBrushing method applied after our Markov walk.

An effective all-round approach would be to use AdultRank followed by the Naïve Bayes classifiers followed by image classifiers.

References

[1] Canny, J.F., "A computational Approach to Edge Detection", IEEE Trans. On Pattern Analysis and Machine Intelligence, Vol. 8, 1986, pp. 679-698.

[2] Fleck, M. M., Forsyth, D. A., and Bregler, C. (1996). "Finding naked people". European Conference on Computer Vision.

[3] Bradski, G. (2000). Programmer's Toolchest:The OpenCV Library. Software at http://opencv.willowgarage.com.

[4] Rowley, H.A, Jing, Y, Baluja, S, "Large scale Image-based adult-content filtering". http://static.googleusercontent.com/external_content/untrusted_dlcp/www.google.com/en/us/research/pubs/archive/38.pdf

[5] Brin, S., Page, L., "The Anatomy of a Large-Scale Hypertextual Web Search Engine", http://infolab.stanford.edu/~backrub/google.html

[6] [Page 98] Page, L., Brin, S., Motwani, R., Winograd, T. "The PageRank Citation Ranking: Bringing Order to the Web". http://google.stanford.edu/~backrub/pageranksub.ps

[7] Jiang, Z., Yao, M., Yi, W., "Filtering objectionable image based on image content", http://portal.acm.org/citation.cfm?id=1758039.

[8] Fei-Fei, L., Fergus, R., Torralba, A., "Recognizing and Learning Object Categories", http://people.csail.mit.edu/torralba/shortCourseRLOC/index.html

Appendix A

Code and some of the data will be made available on http://www.saikatsen.info or http://www.saikatsen.com.

Appendix B

The following custom applications were built during this work on Windows. Some C++ applications require Win64

and need to be compiled in 64-bit mode. Haar classifier requires 64-bit build of open-source OpenCV.

Custom C# apps: (require .Net 2.0)

- Crawler: given a set of sites, crawls them and outputs outbound links. Used to collect list of good and bad sites. This tool lists subdomains as well in the outbound list.
- CLAdultFetcher: gets images from craigslist – positives from Personals, negatives from pets and babies. All images for a post are downloaded. City and section (pets/ babies etc.) are configurable.
- ImageLabeler: tool used to label and classify boxed images (boxed using Office Picture Manager). Images were classified into five classes only two of which were finally used since they had the most samples.
- AdultGuard: crawls sites and extracts title, keywords, url, content; reads a blacklist and graylist file and writes out 1 for word present and 0 for word absent in csv format to an output file that can be fed into the Bayes classifier for training and testing.
- CreateHaarIndexFile: creates an index file for consumption by OpenCV HaarClassifier. The Haar classifier was trained successfully to this index file.
- ImageFeatureExtractor, ImageFeatureExtractorConsole: calculates histogram of gradients, bins them into 16 bins and calculates % in each bin. A second function can apply the Debrush filter to a specified image.
- CreateAdultDictionary: loads the Princeton WordNet dictionary, iterates through words user provides, lets user bucket them into blacklist and graylist and continues iterating. In the end, lets you save the dictionary to files adult.txt and gray.txt. Requires pre-install of WordNet.
- DataStructures: custom library with trie implementation. Used by other apps.
- FileRandomizer: given a file, shuffles the lines in the file. This is needed so that positive and negative data in a file is randomly shuffled into an output file that can be partitioned into train and test data by classifiers. OpenCV allows you to specify a partitioning index between train and test data in a matrix. Without the shuffling, all test data would be negatives.

Custom C++ apps:

- SampleApp::ImageFeatureExtractorHSV: calculates Hue and Hue+Saturation color histograms, applies Canny, Laplace filters.
- SampleApp::Classifier: The master program that runs Random forest classifier, Boosting classifiers, Naïve Bayes and MLP on train and test data, and reports test and train recognition %rate and TP, FP, TN, FN data.

Machine Learning Code, Image Recognition Code:

- OpenCV (http://opencv.willowgarage.com)