Patch-Based Image Classification Using Image Epitomes

David Andrzejewski
CS 766 - Final Project

December 19, 2005

Abstract

Automatic image classification has many practical applications, including photo collection organization and image search. In this project I approach the image classification task through the use of image epitomes. The epitome of an image is a probabilistic model that serves as a compact representation of the shape and texture information present in the original image. I first create a collage of positive and negative example images, then I generate the epitome representation of this collage. Patches in the epitome which are much more likely to be found in positive images than in negative ones can then be used to classify new images.

1 Problem statement

Given sets of positive and negative example images, develop a classifier that will be able to classify new test images as either positive or negative.

2 Image epitomes

Image epitomes were originally proposed in [3]. The epitome of an image consists of a smaller bitmap and a set of mappings from patches in the bitmap to patches in the original image. In this way, the original image can be reconstructed from the epitome, although not perfectly. The epitome

model is useful for multiple vision applications, including segmentation and denoising.

Each pixel in the epitome is modeled as a mixture of gaussians, one for each color channel. For each patch in the original image, a posterior distribution over all patches in the epitome is calculated to represent the possible mappings. To learn the epitome, the mapping probabilities are initialized along with the epitome pixel means and variances. The EM algorithm is then applied to update these values for either a set number of iteration or until convergence.

An essential aspect of the epitome is that, given an image patch in the original image and the image epitome, one can calculate the posterior probability of the mapping from each patch in the epitome to the image patch. The epitome patch with the highest posterior probability of mapping is then used to generate that image patch in the reconstruction. A single epitome patch can, and often does, generate many different patches in the original image. This one-to-many mapping captures textual and shape regularity in the original image.

The website [4] contains more information about image epitomes, as well as example images and MATLAB code. The provided MATLAB code comes in the form of two functions

```
[e,ev,p,etr] = epitome(x,K,N,T,NIT,sc,e,ev)
z = reconstruct(x,K,e,ev,emod,SPC)
```

The epitome function does NIT iterations of EM to learn an N by N epitome of image x using T patches of size K by K. e and ev are optional parameters that supply the initial values for the epitome. sc is another optional parameter that can be used to specify different scaling factors - the EM algorithm is repeated NIT times at each scale, and the end result is used to initialize the epitome for the next scale. In order to save time, I did not use the scaling parameter in this project.

The reconstruct function uses the original epitome (e,ev) along with the original image x to reconstruct the original image with a modified epitome $emod$. This is useful for tracing pixels from the epitome back into the original image. This is well illustrated in the PowerPoint available from [4]. The PowerPoint contains an image of a dog sitting on some gravel in front of flowers. The pixels corresponding to the gravel in the epitome are then colored bright teal, and this modified epitome is used to reconstruct the

original image, resulting in a reconstructed image where the gravel area is bright teal.

One important aspect of the reconstruction algorithm is that the mapping from the epitome to each patch in the original image is "winner take all". That is, the epitome patch with the highest posterior probability of mapping into an image patch is the only one used to generate that patch in the original image. Patches in the original image are allowed overlap, and in this case their contributions to a given pixel are simply averaged.

My project uses these two functions to learn epitomes and to reconstruct images. I also use code fragments from these methods for some epitome-related tasks, like determining which epitome patch has the maximum posterior probability of mapping into a given image patch.

The image epitome concept was extended to video data in [1]. The extension is a natural one and involves modeling video data as a 3-dimensional cube where individual frames from the video are stacked on top of one another along the temporal axis. The video epitome then contains 3-dimensional video cubes instead of 2-dimensional image patches. One of the authors has video examples and code available on the web at [5]. While my project does not explicitly use video epitomes, these resources were useful in gaining a better understanding of the epitome model in general.

3 My approach

My approach to image classification exploits the epitome representation to find patches that are effective at discriminating between positive and negative example images. The approach is as follows

3.1 Train classifier

1. Create a training collage of positive and negative example images.
2. Learn the epitome of the training collage.
3. Calculate the conditional probability of each epitome patch being mapped into an image given that the image is a positive (or negative) example.

4. Select the epitome patches that maximize the ratio

$$\frac{P(patch_i = 1|pos)}{P(patch_i = 1|neg)} \tag{1}$$

5. These patches, and their conditional probabilities of being mapped into positive and negative example images, can then be used to classify new images.

The conditional probabilities of a patch mapping into an image given that the image is a positive example is simply calculated

$$P(patch_i = 1|pos) = \frac{N_i + \alpha}{N + \alpha} \tag{2}$$

where $patch_i = 1$ means that $patch_i$ does map into the example image, N_i is the number of positive example images that patch i maps into, and N is the total number of positive example images. α is a psuedocount to avoid getting zero probabilities. We say that $patch_i$ from the epitome maps into an image if $patch_i$ has the highest posterior probability among all epitome patches of mapping into an image patch for ≥ 1 image patch. The patch mapping probability for negative example images is calculated in a similar fashion.

3.2 Classify new test images

1. For each patch in the epitome, determine whether or not it maps into the test image.

2. Use Bayes Rule and the patch mapping conditional probabilities to calculate the odds ratio of

$$\frac{P(pos|patches)}{P(neg|patches)} = \frac{\prod_i P(patch_i|pos)}{\prod_i P(patch_i|neg)} \tag{3}$$

   where i ranges over the indices of all patches being used to classify.

3. Classify the test image using some threshold for the odds ratio.

The general classification decision framework and the Bayesian odds ratio calculation are very similar to those used in [2].

4 Experiments

4.1 Experimental procedure

In order to test the effectiveness of this classification method, I created 2 datasets each consisting of 13 positive example images and 13 negative example images. For one dataset, the 13 positive example images were close-up images of a single human face, and for the other the 13 example images were all pictures taken at beaches. For both datasets, the negative examples were a haphazard jumble of images that did not fit the description of the positive dataset. All images were obtained from either Google Image Search or Flickr.

To do a single experimental run, I randomly selected an equal number of positive and negative images to be training examples and used them to construct the training collage. The remaining example images were then classified using the discriminative patches extracted from the training collage. This procedure was repeated 3 times for each of the 2 datasets. Running the experiments took between 12 and 13 hours for the faces dataset and about 15 hours for the beaches dataset. The difference is due to the fact that I used more images for training on the beaches dataset, since the classifier performed much worse on the beaches set than on the faces set when using the same number of training examples.

4.2 Experimental parameters and setup

In order to make the training collage easier to construct, all images were resized to be the same size (150 by 113). The epitome parameters K,N,T were determined empirically by preliminary trial-and-error experiments. For the image and training collage sizes we used, setting K=15, N=75, and T=1412 seemed to work well in simple trial experiments. For the faces dataset, I trained on 5 positive and 5 negative examples, then tested on the remaining 8 positive and 8 negative examples. For the beaches dataset, I trained on 8 positive and 8 negative examples, then tested on the remaining 5 positive and 5 negative examples. Originally, the classifier performed very poorly (around 30% accuracy on positive test images) on the beaches dataset when using only 5 pairs of training images.

4.3 Experimental results

The first dataset consists of 13 images that contain a human face, and 13 images that do not. On this dataset the classifier correctly classified approximately 95.8% of the positive examples and 75% of the negative examples. Adjusting the odds ratio classification threshold did not have much of an effect on the results - images tended to get odds ratio scores that were either very high or very low. For this data set, the incorrectly classified negative images were somewhat, but not totally, consistent across experiments. As one would expect, the negative images that were misclassified tended to contain coherent flesh-toned objects, such as a beige wildcat or a sandstone house.

The second dataset consisted of 13 images of scenes on the beach, and 13 non-beach images. On this dataset the classifier correctly classified about 53% of the positive images and 86.7% of the negative images. Again, the threshold did not seem to matter much. The poor performance on positive example images was somewhat surprising. I had selected beaches because I thought the borders between sky and sea and sand would provide excellent discriminative patches. From examining the misclassified positive example images, it seemed that the discriminative patches selected from the training collage epitome were very "picky" about the exact color of the sand, sea, and sky. Beach images under darker illumination were misclassified often, even if there were dark beach scenes in the training set. This seemed to be because the dark training images contributed only sky patches to the discriminative patch set instead of sand patches. The training image clustering modification suggested below in the "Possible Improvements" section may be a potential answer to this problem. On the other hand, the classifier did a very good job at deciding what was not a beach. This may be due to the same "pickiness" over sand, sea, and sky color that caused it to perform poorly at deciding what was a beach.

5 Analysis of the approach

5.1 Approach drawbacks

There are several problems with the patch-based classifier. First, as a completely appearance-based classifier it can be very sensitive to changes in pose, illumination, and scaling. The only way the classifier can handle these types of variations is to include training examples that capture the variation, which

is an inherently inflexible workaround strategy. Second, the classifier uses patches in a "bag of words" strategy that discards any information about the spatial distribution of the patches in the training example images. This is a significant difference from the feature representation used in [2]. Finally, it is also inconvenient to add new training examples to the classifier. With the current classifier setup, in order to take a new training example into account it would be necessary to create a brand new training collage and re-learn the epitome.

5.2 Possible improvements

There are several ways that one could potentially improve the performance of this classifier system. One potential modification would be to take positive training example clusters into account. It may be that a significant subset of the training images share a discriminating epitome patch, but the rest do not. In this case, that epitome patch would not score highly overall for its ability to discriminate between positive and negative training example images, even though it is an excellent classifier patch for a subset of the positive training examples. A practical example would be human faces under low illumination versus human faces under normal illumination. Patches that identify faces under low illumination could be missed because they do not have a high probability of mapping into positive images overall. Checking for patches that are excellent discriminators for a subset of the positive training examples could result in a more robust classifier.

Also, the "winner take all" nature of the procedure for determining whether an epitome patch gets mapped into an image or not could be altered. In the current implementation, only the epitome patch with the highest posterior probability of being mapped into a given patch in the original image is considered. This is obviously discarding information about other patches that had high, but not highest, probabilities of mapping into this image patch. Allowing "partial" mappings weighted by relative posterior probabilities may help retain some of this information.

A scheme for capturing information about the relative spatial distributions of image patches mapped to by discriminative epitome patches may also improve performance. An example of such a scheme can be found in [2].

Finally, the image epitome representation can also be modified in several ways. The results shown in [3] were created using a modified epitome learner that allowed varying block sizes, which allows tesselations of coarser and finer

blocks. The results in [3] show a clearly superior preservation of shapes and structures from the original image. This may enhance the performance of our classification application. Also, [3] mentions the fact that block mappings from the epitome to the image need not be limited to simple copying operations. It would be possible to allow blocks to be scaled, rotated, or otherwise transformed in the mapping process. This modification to the image epitome model seems likely to result in a more robust patch-based classifier.

6 Conclusion

Patch-based image classification with image epitomes showed some success at correctly classifying test images. It performed well on positive example identification on the face dataset and negative example identification on the beach dataset. But there are clearly problems with the technique, as it performed quite poorly on positive example identification on the beach dataset and somewhat poorly on negative example identification on the face dataset. Many of the problems with the approach seem to stem from the somewhat "brittle" nature of the patch-based approach. Making the epitome model more flexible by allowing varying patch sizes and transformational mappings may improve the performance considerably. The epitome representation captures textural and shape regularities present in image data, and the results of my experiments suggest that it may be possible to successfully exploit this property to find features that can be effectively used for image classification.

References

[1] V. Cheung, B. J. Frey, and N. Jojic, Video epitomes, Proc. IEEE Conf. Computer Vision and Pattern Recognition, 2005.

[2] R. Fergus, P. Perona, A. Zisserman, Object Class Recognition by Unsupervised Scale-Invariant Learning, Proc. of the IEEE Conf on Computer Vision and Pattern Recognition, 2003.

[3] N. Jojic, B. J. Frey, and A. Kannan, Epitomic analysis of appearance and shape, Proc. 9th Int. Conf. Computer Vision, 2003.

[4] http://research.microsoft.com/∼jojic/epitome.htm

[5] http://www.psi.toronto.edu/∼vincent/videoepitome.html

README

This section explains the some of the supplemental materials attached to the physical handin of this final report.

Experimental results

- Result images = These are the resulting images from the experimental runs
- Result report = This is the automatically generated text report of the experiments

MATLAB code used in project

- epitome.m = Function to learn the epitome from an image, obtained from [4]
- reconstruct.m = Function to reconstruct original image from the (possibly modified) epitome, obtained from [4]
- crossValidate.m = Top level script - handles experimental setup, reads in images, randomly selects training/testing sets, calls experimentRun for each iteration, and organizes reporting of experimental results
- experimentRun.m = Given training and testing image sets, run a single classification experiment
- patchTrace.m = Reconstruct the original image, painting green borders around patches in the image that were generated by discriminative patches in the epitome - this code is a slightly modified version of reconstruct.m from [4]
- classify.m = Given an image, the mapping status for the discriminative patches(do they map into this image or not?), the conditional mapping probabilities $P(patch_i|pos)$ and $P(patch_i|neg)$ for each patch i, calculate the classification odds ratio $\frac{P(pos|patches)}{P(neg|patches)}$
- getBest.m = Returns the coordinates of the k top-scoring epitome patches, where k is a parameter

- isPatchMappedAll.m = Given an image and an epitome, for each patch in the epitome determine whether or not it is mapped into the image

MATLAB code developed during project but not used for final results

- getPrecRec.m = Function to calculate precision and recall values for varying classification threshold values - used to determine that test images were either getting very high odds ratios or very low ones, so the threshold does not make a significant difference
- findCritical.m = Calculates the positive/negative odds ratio of the average max posterior mapping probability for each epitome patch over all training examples (this was my previous method for discriminative patch identification - this code is not used in final implementation)
- findMaxPost.m = For an epitome patch and an example image, find the max posterior probability of this epitome patch mapping into this image (helper function used by findCritical)
- calcPosterior.m = Given a patch in the original image, calculate the posterior probabilities of all epitome mappings into this patch (helper function for findMaxPost)

It is also important to note that many of the MATLAB functions contain code fragments from the epitome.m and reconstruct.m files obtained from [4]. I have attempted to document these instances in the handin code by clearly marking them with green ink and the letters "MSR" (for **M**icro**S**oft **R**esearch).