Image Classification using Latent Patch Concepts

Tushar Khot
Computer Sciences Department
University of Wisconsin, Madison, WI
tushar@cs.wisc.edu

Abstract

There have been many approaches to tackle the problem of image classification. Some sacrifice speed for a very complex model whereas other approaches try to limit the features for speed. In our approach, we try to maintain spatial information similar to the spatial pyramid approach and use the pyramid match kernel for efficiency. We split the images into overlapping patches at multiple scales and train a latent concept model on these patches. For testing, we find the most likely category using a naive latent topic model. Patches of different scales have different classifiers, which can be utilized for parallelizing the process.

1 Introduction

The problem that we are attempting to solve is the image categorization problem over the Caltech-101[1] and Caltech-256[2] dataset. Over the last few years, it has been shown that simpler approaches actually work really well for image classification[5]. Using a simple histogram intersection kernel with SVM actually performs really well as shown in [4]. The spatial pyramid kernel used in [4] splits the image into partitions and builds a histogram over all the partitions. By using the partitions they take into account the location of the features too. Extending this idea, we built an image classification tree where nodes at higher depths deal with smaller partitions of the tree. Each node of the tree deals with a particular partition of the image and learns a classifier based on these partitions in all the images. Also if the SVM/KNN has a very high classification probability, we can completely avoid traveling down the tree. Unfortunately, the recognition performance of this approach was very poor as it didn't allow for matches across patches at different positions. Hence we then decided to use latent topic models, to allow for matches between patches irrespective of their position.

2 Related Work

Grauman et al. [3] used a pyramid kernel, by splitting the feature space into partitions. But their approach didn't account for the location of these features.

Lazebnik et al. [4] extended this idea but instead of splitting the feature space, they split the image into parts and built a histogram intersection kernel on the features.But the objects in every image may not be at the same location and hence this approach may not always work.

There have been many other kernels such as [11] which can then be used with KNN or SVM's but most of these approaches are too time consuming and hence were not used.

There have been approaches that use other machine learning techniques such as decision trees [8] or SVM/KNN [6] for classification.

Boiman et al. [5] have shown that using complicated features that try to reduce the size of the feature space lose a lot of information and hence they use a simple Nearest-Neighbor approach. We also rely on the simplicity of our approach to make use of the most information without taking too much time.

Desai et al.[9] use object detection to solve the image categorization problem. But they need the objects in the training data which may not always be available. Instead latent topic models,[10], [7] are more appealing as they don't rely on the objects

being provided in the training data.

3 Implementation

Since, we wanted to match patches irrespective of their location, we needed to train a classifier on all the patches of the same scale. Since nearest neighbor approaches lose the least amount of information, we decided to use K-Nearest Neighbor for classification. To reduce the search time during testing, we also performed clustering on the training patches. The cluster centers formed the latent topics for the classification problem. The probability of an image can thus be calculated by computing the probability of its patches being generated from some concept and the probability of that concept belonging to the given category.

3.1 Training

Every training image is partitioned into overlapping patches at various scales.In our approach, a scale of 1/2 indicates that the patches have half the rows and columns as compared to the original image. Given the boundaries of these patches, we can find the PCA-SIFT features enclosed within these boundaries. The set of PCA-SIFT features for each window, form the feature patches.

All patches at the same scale but of different categories are collected together. We perform Hierarchical Agglomerative Clustering on these patches to find the concepts as centroids. We use the pyramid match kernel to find the pairwise distance between all the pair of feature patches(set of PCA-SIFT features). The pair of points with the highest similarities are clustered and the centroids are added back to the list of patches. For clustering two feature patches, we perform KMeans clustering on the collection of features within these patches. We then select the set of PCA-SIFT features from the original set nearest to the center. We don't use the centroids directly as the pyramid match kernel uses a vocabulary tree on the original set of PCA-SIFT features and fails on seeing unknown features.

To save on time, we cluster twenty patches at a time before recomputing the distances. We stop clustering once the number of clusters are small enough or the similarity measures are too low for any meaningful clustering.

The distance computation is skewed by the fact that some patches may have very high number of features and hence the intersection would also be very high. Hence we divide the similarity score by the number of features in each patch.

Also, images of certain classes are inherently sparse and would always have low similarity measures. We cluster all patches with less than 5 features into a default *blank patch*.

At the end of the HAC run, we have a set of new patches where each patch, L_j was obtained by combining n_{jk} patches from category C_k.

$$Prob(C_k|L_j) = \frac{n_{jk} + m}{\sum_k (n_{jk} + m)}, \text{where m=0.1} \quad (1)$$

Algo 1 Train(Images, Features, Categories)

1. Perform the following for windows at scale, $s = \{1, 1/2, \cdots 1/2^n\}$.
2. Partition the image and the features into overlapping windows at scale s.
3. Perform HAC on the feature patches across categories at scale s, till number of patches < threshold.
4. For each cluster/concept, maintain the $P_s(Category|Concept)$.

3.2 Testing

At the end of training phase, we have a set of latent concepts and their corresponding probabilities. For testing on a new image, we perform the same partitioning of an image into feature patches. For every feature patch, we find the nearest neighbors in the latent concepts at the same scale.For the patches with less than 5 features, *blank patch* is their default and only neighbor.

The similarity scores for all patches except the top k, are ignored and assumed to be 0. Given the similarity between patch, P_j and concept patch, L_k as d_{jk},

$$Prob_d(L_k|P_j) = \frac{d_{jk}}{\sum_k d_{jk}} \quad (2)$$

Now, we compute the probability of each category for every patch. We assume that every patch influences the category independently. Using this Naive Bayes assumption, we get

$$Prob_d(C_i|P_j) = \sum_k Prob(C_i|L_k) * Prob(L_k|P_j) \quad (3)$$

$$Prob_d(C_i|P_1, P_2, \cdots P_n) = \prod_j Prob(C_i|P_j) \quad (4)$$

These probabilities are only applied at a given scale and we need to combine the probabilities at multiple scales. Similar to spatial pyramids, we also increase the weight as the size of the patches reduce. Considering all patches at the scale of 1 at depth 0, patches at scale 1/2 at depth 1 and so on.

$$Prob(C_i) = \sum_{d=1}^{depth} \frac{1}{2^{depth-d+1}} Prob_d(C_i|P_1, \cdots P_n) \quad (5)$$

Algo 2 Test(Image, Features, Classifiers)

1. For each scale s $\in \{1, 1/2, \cdots\ 1/2^n\}$.
2. Create feature patches of scale s from the input image.
3. For each patch, P_j
4. Find top k nearest neighbours in the training concepts.
5. Compute $P(Concept|P_j)$ for these concepts for every patch.
6. Compute $P(Category|P_j)$ using the training probabilities.
7. Compute $P(Cateogry|P_1, P_2, \cdots P_n)$.
8. Combine the probabilities at all scales, and select the category with the highest probability.

4 Experiments

We have tested our approach on Caltech256 dataset and Caltech101 dataset. Unless otherwise mentioned, all the experiments have been trained on 30 images and tested on 10 images. All percentage numbers are recognition rates which is the recall of our classifier.

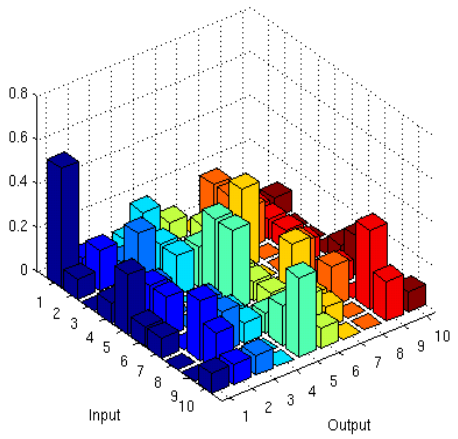

Figure 3: Confusion Matrix for 10 categories

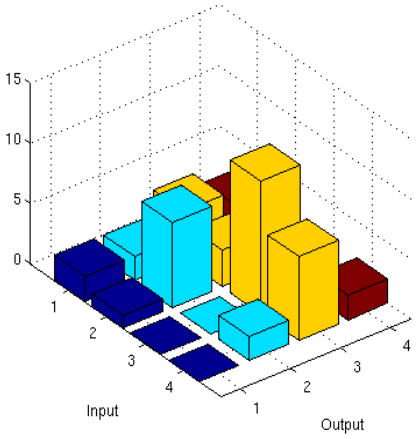

Figure 4: Confusion Matrix for 4 categories

Figure 3 gives the confusion matrix over 10 Caltech-256 categories as a bar graph. The recognition rate was 25.45% which is better than random guessing(10%).

Figure 4 gives the confusion matrix over 4 Caltech-101 categories as a bar graph. The recognition rate was 20% which is worse than random guessing. Our approach didn't seem to work well

Figure 1: Caltech 256 Dataset

Figure 2: Caltech-101 Dataset

with Caltech-101 dataset.

4.1 Impact of overlap

Overlap is the portion of the feature patches that is common with other feature patches too. Increasing the overlap between the feature patches actually has a negative impact on the performance, although the difference is really small(60% to 69%).

```
    0.9091    0.0909         0
    0.1818    0.5455    0.2727
    0.0909    0.2727    0.6364
Recognition Rate:69.69%
```

Confusion Matrix 1: Half of image overlaps

```
    0.9091    0.0909         0
    0.2727    0.3636    0.3636
    0.0909    0.3636    0.5455
Recognition Rate:60.6%
```

Confusion Matrix 2: Two-thirds of image overlaps

4.2 Impact of scale

Scale of a patch is ratio of the number of rows/columns in the patch to the rows/columns in the original image. If we use smaller patches at depth 2, i.e. use patches of scale 1/4 instead of 1/2, the performance doesn't improve but actually drops.

4.3 Impact of depth

Depth indicates how many times we split the patches to form a new set of patches at a smaller

```
    0.9091    0.0909         0
    0.1818    0.5455    0.2727
    0.0909    0.2727    0.6364
Recognition Rate:69.69%
```

Confusion Matrix 3: Patches of scale: 1/2

```
    0.8182    0.1818         0
    0.2727    0.3636    0.3636
    0.0909    0.2727    0.6364
Recognition Rate:60.6%
```

Confusion Matrix 4: Patches of scale : 1/4

scale. At depth=d, the scale of the images is $1/2^{d-1}$. To check whether increasing the depth of the tree has any advantages, we tested our approach using a tree with depth=2 and depth=3. We trained on 30 images of 4 categories and tested on 16 images. The confusion matrices(total counts used instead of percentages) are shown below:

```
15     1     0     0
 7     7     0     2
11     3     0     2
 0     4     0    12
```

Confusion Matrix 5: Tree Depth 3

```
15     1     0     0
 6     6     1     3
 8     2     4     2
 0     4     0    12
```

Confusion Matrix 6: Tree Depth 2

```
10     5     1     0
 5     8     0     3
 6     6     4     0
 0     8     1     7
```

Confusion Matrix 7: Tree Depth 1

As it can be seen that the difference between recognition rate for depth=2 and depth=3 is not much but increasing the depth increases the time by quite a margin. Actually increasing the depth has a negative impact after depth=2. Whereas increasing the depth to 2 from 1, improves the recognition rate from 45% to 57%.

4.4 Impact of training data

As the amount of training data is increased, the performance of the classifier improves.

```
    0.9091    0.0909         0
    0.3636    0.4545    0.1818
         0    0.7273    0.2727
Recognition Rate:60.6%
```

Confusion Matrix 8: Training Set:20

```
    0.9091    0.0909         0
    0.3636    0.5455    0.0909
    0.0909    0.5455    0.3636
Recognition Rate:60.6%
```

Confusion Matrix 9: Training Set:25

```
    0.9091    0.0909         0
    0.1818    0.5455    0.2727
    0.0909    0.2727    0.6364
Recognition Rate:69.69%
```

Confusion Matrix 10: Training Set:30

```
    1.0000         0         0
    0.4545    0.3636    0.1818
         0    0.2727    0.7273
Recognition Rate:69.69%
```

Confusion Matrix 11: Training Set:40

4.5 Impact of Clustering

We cluster the feature patches to about θ% of the input set of patches. Having a high θ would have faster training and slower test performance. Also, since most of the patches are not clustered it would behave like nearest neighbors. On the other hand, smaller percentage would increase the training time

a lot, and would generalize better. But in our experiments smaller threshold percentage worked better.

```
    0.9091    0.0909         0
    0.1818    0.5455    0.2727
    0.0909    0.2727    0.6364
Recognition Rate:69.69%
```

Confusion Matrix 12: Clustering 90%

```
    0.9091    0.0909         0
    0.2727    0.3636    0.3636
    0.0909    0.3636    0.5455
Recognition Rate:60.6%
```

Confusion Matrix 13: Clustering 70%

5 Future Work

Currently the results don't look great and also performing HAC is really slow. Probably using a patch tree similar to a vocabulary tree could speed up search during testing and also prevent unnecessary computations during HAC.

The model that is used currently is very naive and can be improved upon by using models similar to LDA. Probably using object cuts would be useful to find better patches to match.

6 Conclusion

Using spatial information along with pyramid match kernels brings in the good from both the worlds. Clustering among patches helps to reduce the test time and also generalizes across examples. Using soft nearest neighbors also helps to capture richer probability distributions among patches and latent concepts.

References

[1] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories. In *IEEE CVPR Workshop on Generative-Model Based Vision, 2004.* `http://www.vision.caltech.edu/ImageDatasets/Caltech101`.

[2] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical report, CalTech, 2007.`http://www.vision.caltech.edu/Image_Datasets/Caltech256/`

[3] K. Grauman and T. Darrell. Pyramid match kernels: Discriminative classification with sets of image features. In *Proc. ICCV, 2005.*

[4] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR, 2006.*

[5] O Boiman, E Shechtman, M Irani. In defense of nearest-neighbor based image classification. In *CVPR, 2008.*

[6] H. Zhang, A. Berg, M. Maire, and J. Malik. SVM-KNN: Discriminative nearest neighbor classification for visual category recognition. In *Proc. CVPR, 2006.*

[7] Florent Monay, Pedro Quelhas, Jean-Marc Odobez. Contextual classification of image patches with latent aspect models. In *EURASIP 2009.*

[8] R Mare, P Geurts, J Piater, L Wehenkel.Random subwindows for robust image classification. In *CVPR 2005.*

[9] Chaitanya Desai ,Deva Ramanan, Charless Fowlkes. Discriminative models for multi-class object layout. In *ICCV 2009.*

[10] Chong Wang, David Blei, Li Fei-Fei. Simultaneous Image Classification and Annotation. In *CVPR 2009.*

[11] R. Kondor and T. Jebara. A Kernel Between Sets of Vectors. In *Proccedings of ICML, 2003.*