Image Classification using Bag of Words Model

1. Algorithm

1.1 Introduction

In Computer Vision, the Bag of Words model is used for image classification, by treating image features as visual words. Bag of Words is a sparse vector of occurrence counts of visual words; i.e., a sparse histogram over the vocabulary.
The algorithm described here, classifies images from different categories using Bag of Words model and K Nearest Neighbors method. This algorithm uses TF-IDF concepts to transform visual word frequencies in histogram (over vocabulary) into TF-IDF scores. It then uses TF-IDF scores, coupled with query search analogy as a similarity measure in order to find K closest neighbors of test image. Finally, label is assigned to test image based on majority of votes by K nearest neighbors.

1.2 Algorithm

1.2.1 Setup and Initialization

1. Create 'BOW.mat' file to store codebook, histograms and its respective labels, TF-IDF scores.
2. Download VLFeat library from website, extract and copy it to folder where source code is kept.
3. Add VLFeat to MatLab environment.
   e.g.
   ```
   run([VLFEATROOT, 'toolbox/vl_setup']);
   ```
4. Configure Bag of Words model for different parameters (Configuration.m).
   > Path for training, validation and testing images
   > Allowed image types
   > Standard resolution for preprocessed images
   > Parameters for Multi-Scale Dense SIFT features
   > Parameters for HoG features
   > Parameters for K-Means quantization
   > Histogram bin size
   > K Nearest Neighbors

1.2.2 Training

1. Preprocess training images,
   a. Convert training images to single precision.
   b. Standardize images using fixed resolution for all training images.
      e.g. (128, 128, 3)
2. Create a codebook of visual words.
   Note: This algorithm generates codebooks for two different types of features.
      - Multi-Scale, Color, Dense SIFT
      - HoG
   a. SIFT Codebook:
      - Generate Multi-Scale, Color, Dense SIFT local features.
        Note: This algorithm uses wrapper function of vl_sift() i.e. vl_phow() to generate multi-scale, color, dense SIFT local features.

e.g. Set of parameters used with vl_phow():

```
Phow.Sizes = [3 5 7];
Phow.Fast = false;
Phow.Step = 4;
Phow.Color = 'RGB';
Phow.ContrastThreshold = 0.005;
Phow.WindowSize = 1.5;
Phow.Magnif = 6;
```

- Quantize SIFT local features using K-Means algorithm.

  Note: This algorithm uses vl_kmeans() function to apply K-Means algorithm on extracted local features.

  e.g. Set of parameters used with vl_kmeans():

```
KMeans.SIFT.NUMCENTERS = 150;
KMeans.Distance = 'L2';
KMeans.Initialization = 'PLUSPLUS'; % K-Means++ algorithm
KMeans.Algorithm = 'ELKAN';
KMeans.NumRepetitions = 1;
KMeans.MaxNumIterations = 2500;
```

- Save SIFT codebook in 'BOW.mat' file.

b. HoG Codebook:

- Generate Dense SIFT interest points.
- Generate HoG local features around SIFT interest points.

  Note: This algorithm uses extractHOGFeatures() function to extract HoG local features around SIFT interest points.

  e.g. Set of parameters used with extractHOGFeatures():

```
HoG.CellSize = [8 8];
HoG.BlockSize = [2 2];
HoG.BlockOverlap = [1 1];
HoG.NumBins = 9;
HoG.UseSignedOrientation = true;
```

- Quantize HoG local features using K-Means algorithm.

  Note: This algorithm uses the same set of parameters as described above in (2-a). The only difference is between number of cluster points, where it considers 100 cluster points for quantizing HoG local features.
- Save SIFT codebook in 'BOW.mat' file.

3. Generate image histograms over vocabulary of visual words.

   For every training image,

   For every codebook(feature) type,

   a. Extract local features for the given training image.
      e.g. Dense SIFT, HoG, etc.
   b. Assign every extracted local feature to respective closest cluster (visual word).
      Note: Here, algorithm refers codebooks generated in previous step.
   c. Compute visual word frequencies for a given training image.
      Note: This step generates a histogram of visual word frequencies for a given training image.

   Note: Here, we horizontally concatenate histograms generated for different types of codebooks(features).

4. Transform visual word frequencies in histograms into TF-IDF scores.

   a. Find the logarithmic term frequencies for visual words.

      e.g.

```
TF = 1 + log(HISTOGRAM);
```

```
TF(TF < 0) = 0;
```

b. Find document frequencies for visual words.
   Note: Total number of images in which certain visual word incurs.
   e.g.
   ```
   DF = sum(HISTOGRAM~=0, 1)
   ```
c. Find the logarithmic inverse document frequencies for visual words.
   e.g.
   ```
   IDF = log(1 + (Total images / (DF+1)))
   ```
d. Finally, compute TF-IDF scores for visual words in histogram.
   e.g.
   ```
   TFIDF = TF * IDF
   ```

1.2.3 Testing

1. Preprocess testing images using exactly same procedure as specified in Training section.
2. Extract local features from testing image.
   Note: This algorithm extracts below local features as specified in Training section.
   - Multi-Scale, Color, Dense SIFT
   - HoG
3. Generate histogram over vocabulary of visual words for a given testing image.
   Note: This algorithm generates histogram over visual words as specified in Training section.
4. Compare histogram of testing image with histogram of every training image and assign label to test image based on similarity measure.
   Note: Here, algorithm uses TF-IDF scores, coupled with query search analogy as a similarity measure in order to find K closest neighbors of test image.
   a. Compute similarity scores for the test image's histogram with every training images' histogram.
      e.g.
      ```
      scores = TFIDF * TestHistogram'
      ```
   b. Sort similarity scores in descending order.
   c. Find K closest neighbors based on highest similarity scores.
   d. Get the vote of everyone from top K neighbors.
   e. Find the majority of votes and assign respective label to test image.

1.2.4 Accuracy Measurement

The algorithm computes accuracy simply based on total number of correct predictions against total number of test images.

e.g.

```
accuracy = sum(PredictedLabel==OriginalLabel) ./ Total test images
```

2. Experiments

The experiments were done with described algorithm for classifying 600 test images of 30 different categories (20 images from each category).

2.1 Training

2.1.1 Statistics

Parameter	Value
Training images	1800 images from 30 different categories (60 images from each category)
Codebook Types used	1. Multi-Scale, Color, Dense SIFT 2. HoG
Standard resolution	[128, 128]
K-Means Cluster Points	250
K Nearest Neighbours	11

2.1.2 Log

```
>>
    Started training.....
    Generating Codebook
    kmeans: Initialization = plusplus
    kmeans: Algorithm = Elkan
    kmeans: MaxNumIterations = 2500
    kmeans: MinEnergyVariation = 0.000100
    kmeans: NumRepetitions = 1
    kmeans: data type = double
    kmeans: distance = l2
    kmeans: data dimension = 384
    kmeans: num. data points = 4237200
    kmeans: num. centers = 150
    kmeans: max num. comparisons = 100
    kmeans: num. trees = 3
    kmeans: repetition 1 of 1
    kmeans: K-means initialized in 167.37 s
    kmeans: Elkan iter 0: energy = 2.87362e+07, dist. calc. = 577100707
    kmeans: Elkan iter 1: energy <= 1.85364e+07, dist. calc. = 294986237
    ........
    kmeans: Elkan iter 1069: energy <= 1.73999e+07, dist. calc. = 40749
    kmeans: Elkan terminating because the algorithm fully converged
    kmeans: Elkan: total dist. calc.: -466873665 (0.56 % of Lloyd)
    kmeans: K-means terminated in 3855.71 s with energy 1.73671e+07
    kmeans: Initialization = plusplus
    kmeans: Algorithm = Elkan
    kmeans: MaxNumIterations = 2500
    kmeans: MinEnergyVariation = 0.000100
    kmeans: NumRepetitions = 1
```

kmeans: data type = double
kmeans: distance = l2
kmeans: data dimension = 36
kmeans: num. data points = 4134600
kmeans: num. centers = 100
kmeans: max num. comparisons = 100
kmeans: num. trees = 3
kmeans: repetition 1 of 1
kmeans: K-means initialized in 11.78 s
kmeans: Elkan iter 0: energy = 944352, dist. calc. = 294765863
kmeans: Elkan iter 1: energy <= 686406, dist. calc. = 82235463
....
kmeans: Elkan iter 1519: energy <= 628328, dist. calc. = 13594
kmeans: Elkan terminating because the algorithm fully converged
kmeans: Elkan: total dist. calc.: 2121811798 (0.34 % of Lloyd)
kmeans: K-means terminated in 2520.17 s with energy 624446
Generating Train Histograms
Generating TF-IDF Scores
Training completed.....
>>

2.2 Validation

2.2.1 Accuracy

0.3917 (39.17%)

Note: This accuracy is greater than that of baseline module (~25%)

2.2.2 Log

>>
Started testing.....
Generating Test Histograms
Applying K Nearest Neighbours
Accuracy: 0.3917
Testing completed.....
>>

3. Improvements

3.1 Multiple runs of K-Means

K-Means can be stuck in local optima.
So, K-Means can be run for more than once, e.g. 5 and the one with the best clustering can be selected. This way, algorithm will avoid any local minima.

3.2 More feature types

Many other types of features can be considered to create a Bag of Word model. E.g. SURF, Color Histogram, etc.

3.3 More cluster points

In demo run, algorithm uses 250 cluster points. Cluster points can be increased in order to get bigger distribution in histogram and hence more accuracy.

3.4 Hierarchical Clustering

Hierarchical clustering can be used to increase speed of algorithm.

4. References

1. http://www.vlfeat.org/index.html
2. https://en.wikipedia.org/wiki/Bag-of-words_model_in_computer_vision

5. Repository

https://github.com/chetanborse007/Image-classification-using-BOW