ECE 661 (Fall 2016) - Computer Vision - HW 8

Debasmit Das

November 8, 2016

1 Local Binary Pattern Feature Extraction

The Local Binary Pattern Feature Extraction method has the following steps -

- **Characterize the local inter-pixel variations in gray levels by binary patterns.** We consider neighboring pixels by the equation $(\Delta u, \Delta u) = (Rcos(\frac{2\pi p}{P}), Rsin(\frac{2\pi p}{P}))$ $p = 0, 1, 2...P-1$. We can choose $R = 1$, $P = 8$. The gray levels at the neighborhood pixels (x), of which we have 8 when P = 8, must be computed with an appropriate interpolation formula. We choose the bilinear interpolation formula. Let A, B, C, D be the 4-neighbors of the neighboring pixel x. By bilinear interpolation, the intensity is given by $I(x) \approx (1-\Delta u)(1-\Delta v)A + (1-\Delta u)\Delta vB + \Delta u(1-\Delta v)C + \Delta u\Delta vD$. After estimating gray levels at each of the P points on the circle, we threshold these gray levels with respect to the gray level value at the pixel at the center. If the interpolated value at a point p is equal or greater than the value at the center, we set that point to 1. Otherwise, we set it to 0.
- **Generating Rotation-Invariant Representations**. This is done so that the representations become invariant to in-plane rotation. For each binary pattern, we convert it into an equivalent binary pattern such that this equivalent pattern is that circular permutation of the original pattern, that converts into the minimum integer form in the decimal system. We implement this in MATLAB using an exhaustive search of all the circular permutations of the original binary pattern.
- **Encoding the minimum integer form of the Local Binary Pattern**. We want to use the minimum integer form of binary representation (minIntVal) at all pixels to form a representation of the image. For that, we have to use a histogram. However, We consider those minIntVal patterns as significant, those consists of single run of 0's and 1's. We call these patterns as uniform. If we have P bits for encoding we follow these rules -
  - if the minIntVal representation of a binary pattern has exactly two runs, represent the pattern by the number of 1s in the second run. Such encodings would be integers between 1 and $P1$, both ends inclusive
  - if the minIntVal representation consists of all 0s, we represent it be the encoding 0.
  - if the minIntVal representation consists of all 1s, we represent it by the encoding P.
  - if the minIntVal representation involves more than two runs, encode it by the integer $P+1$.

  So, we have a histogram containing $P+2$ bins. The histogram is finally normalized to make it invariant to the size of the image.

So, for each training image we extract the LBP histogram to form the $P+2$ feature vector to be used in the k-Nearest Neighbors algorithm.

2 Nearest Neighbor Classifier

In pattern recognition, the k-Nearest Neighbors algorithm (or k-NN for short) is a non-parametric method used for classification. In both cases, the input consists of the k closest training examples in the feature space. In k-NN classification, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor. We apply this algorithm on the LBP feature space and use euclidean distance as the distance metric. For our experiments we set $k = 5$. This algorithm is powerful in certain cases -

- Training set size is less
- Training examples are representative enough
- The feature space is chosen such that it is very easy to discriminate among the various classes

In our assignment, the classes are chosen such that they are discriminative in terms of texture. So we use a texture-based representation for images. In our assignment, the classes are chosen such that they are discriminative in terms of texture. So we use a texture-based representation (LBP) for images.

3 Observations

We have the following observations - Overall the accuracy is **70**% which is justified as follows -

- The mountain has the most discriminative of all the LBP histogram features. Accordingly, it has a perfect accuracy of 100% (See confusion Matrix in Experiments section)
- In the case of target class building, some of them have been wrongly labeled as cars. This is because, texture-wise they are quite similar (Cars and Buildings both have windows).
- Cars on the other hand are not wrongly labeled as buildings because they have been distinctive textures of wheels
- Trees are sometimes classified as buildings and mountains because they have rugged texture on their body

4 Experimental Results

We set the following parameters -

$R = 1$ (Radius of the local neighborhood)
$P = 8$ (Number of points on the circle boundary of the neighborhood)
$k = 5$ (Number of neighbors for k-nearest neighbor's algorithm)
Accuracy comes out to be 70%

4.1 Histogram

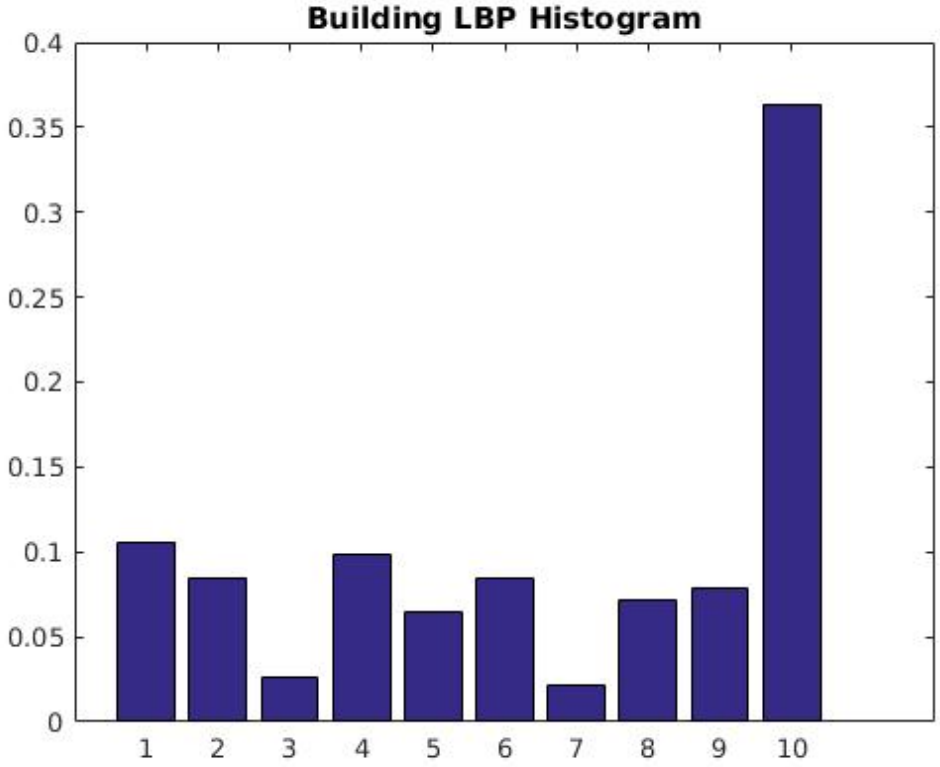

Figure 1: For example 1 of building car training data

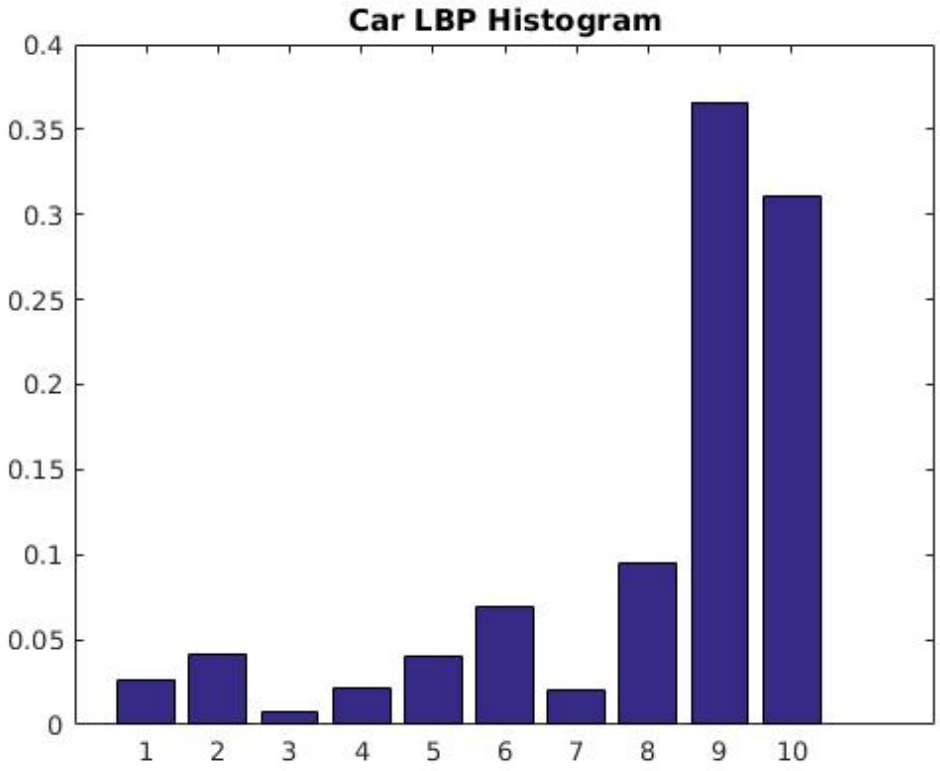

Figure 2: For example 1 of car training data

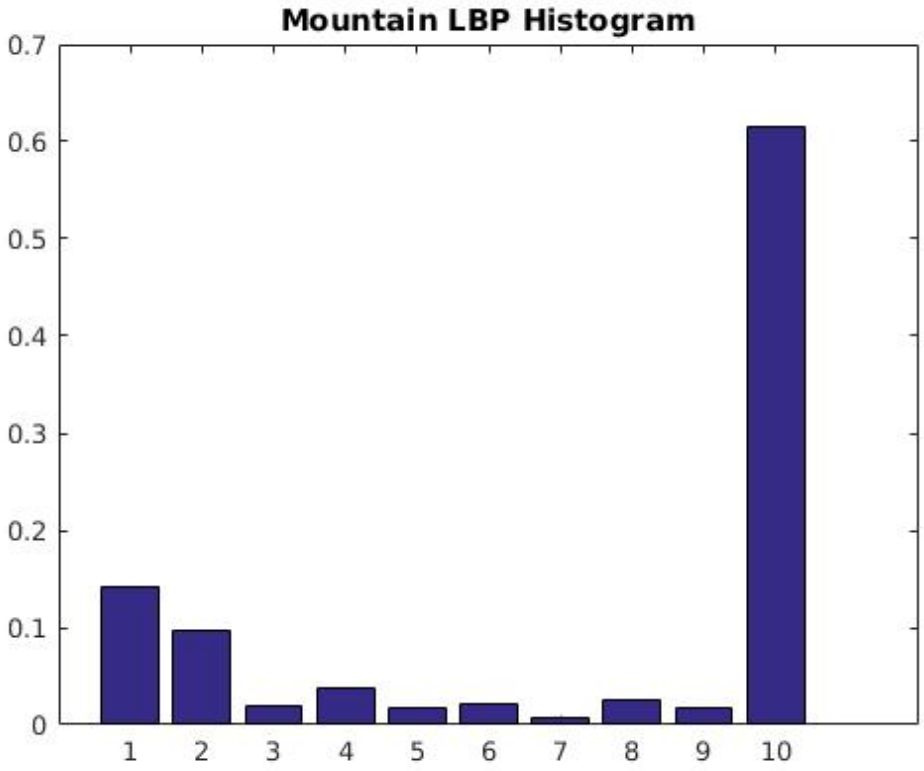

Figure 3: For example 1 of mountain training data

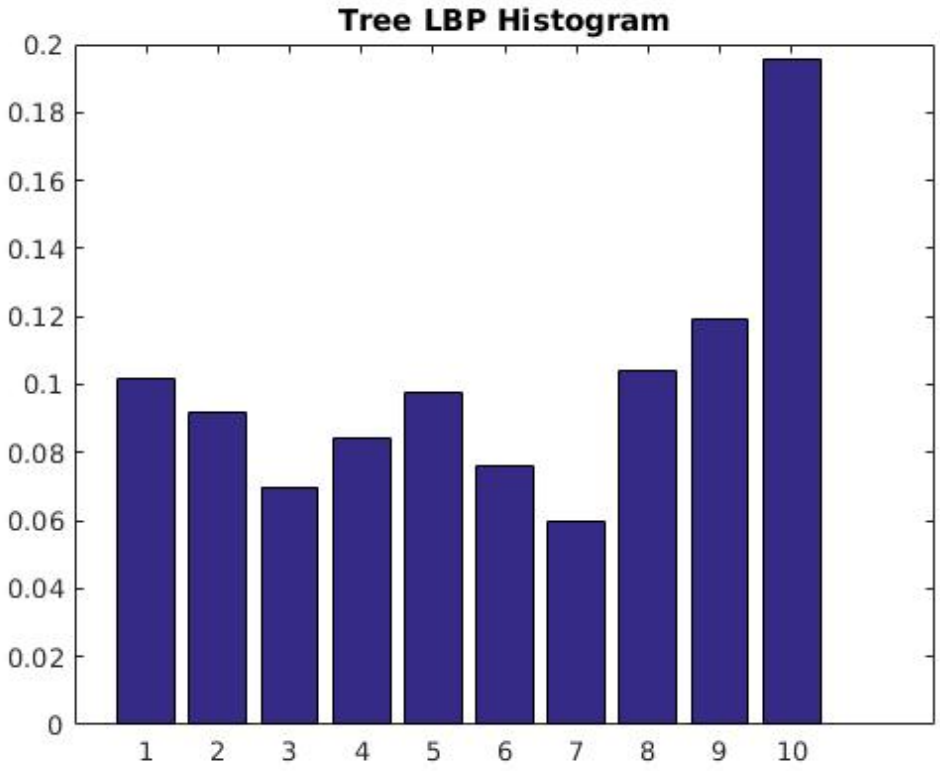

Figure 4: For example 1 of tree training data

4.2 Confusion Matrix

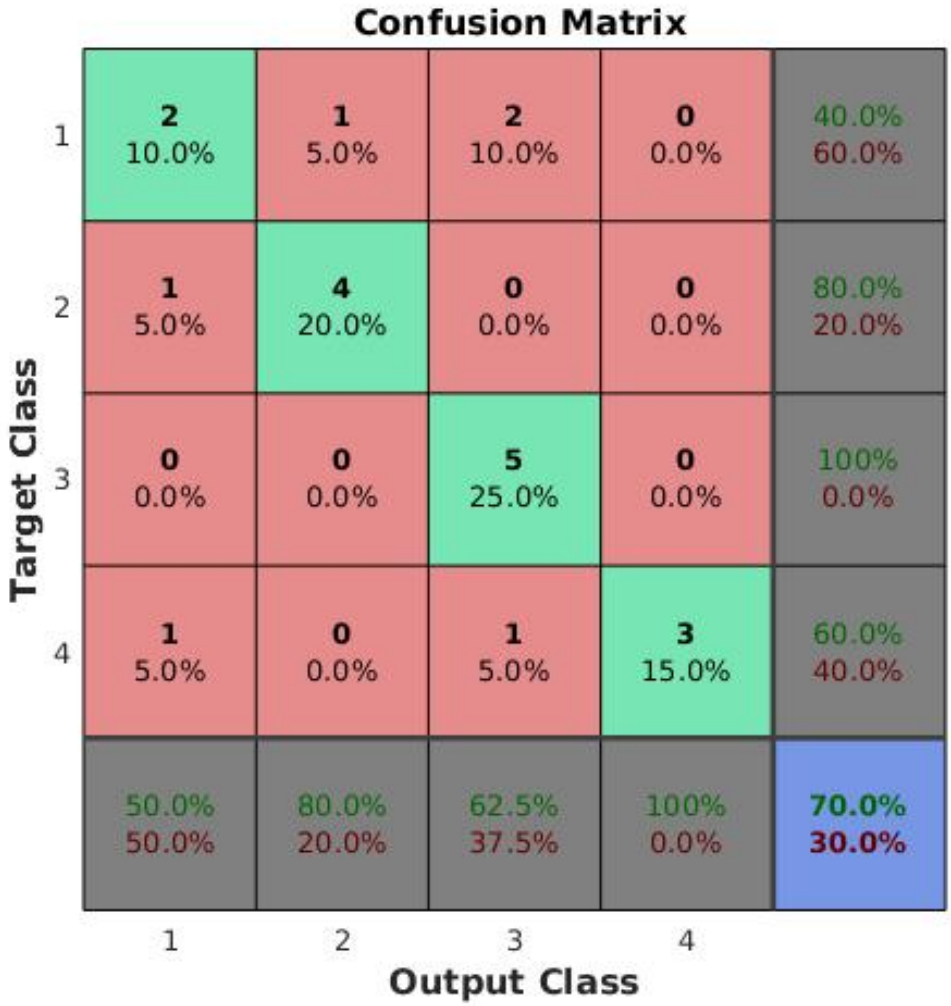

Figure 5: Confusion Matrix for the Classification

Code

The script is in MATLAB 2016a and is self-explanatory

minIntVal Script

```
function pout=minIntVal(pin,P)
A=size(P,1);
minshift=P+1;
for i=1:P
    %Looking over all circular shifts
    Y=circshift(pin,i);
```

```
    A(i)=bi2de(Y','left-msb');
end
[~,minshift]=min(A); % Finding the index of the one with minIntVal

pout=circshift(pin,minshift);
end
```

Local Binary Pattern Extraction Script

```
function f=LocalBP(img,R,P)
% The output f is the LocalBP feature representation
% The input img is the image
% R is the radius of the circular pattern
% P is the number of points to sample on the circle
imgg=rgb2gray(img);
f=zeros(P+2,1);

for i=R+1:size(imgg,1)-R
    for j=R+1:size(imgg,2)-R
        patt=[];


        for p=1:P
            % Neighbours are calculated
            delk=R*cosd(360*p/P);
            dell=R*sind(360*p/P);
            % To do Bilinear transformation
            k=i+delk; l=j+dell;
            kbase=floor(k);  lbase=floor(l);
            deltak=k-kbase; deltal=l-lbase;
            if (deltak==0 && deltal==0)
                imgp=img(kbase,lbase);
            elseif (deltal==0)
                imgp=(1-deltak)*imgg(kbase,lbase) + deltak*imgg(kbase+1,lbase);
            elseif (deltak==0)
                imgp=(1-deltal)*imgg(kbase,lbase) + deltal*imgg(kbase,lbase+1);
            else
                imgp=(1-deltak)*(1-deltal)*imgg(kbase,lbase) + ...
                     (1-deltak)*deltal*imgg(kbase,lbase+1) + ...
                     deltak*deltal*imgg(kbase+1,lbase+1 ) + ...
                     deltak*(1-deltal)*imgg(kbase+1,lbase);
            end

            % Pattern is updated
            if (imgp >=imgg(i,j))
                patt=[patt;1];
            else
                patt=[patt;0];
            end
        end
```

```
        minpatt=minIntVal(patt,P);
        v=minpatt';
        % Code to find the number of runs of zeros and ones
        w=[1 v 1];
        runs_zeros = find(diff(w)==1)-find(diff(w)==-1);
        number_runs_zeros = length(runs_zeros);
        number_runs_ones = number_runs_zeros-1+v(1)+v(end);

        % Proper Encoding is done
        if ((number_runs_zeros + number_runs_ones)> 2)
            f(P+2)=f(P+2)+1;
        else
            f(sum(minpatt)+1)=f(sum(minpatt)+1)+1;
        end
    end
end

f=f/sum(f);
end
```

Main Script

```
nneighbors=5; % no. of neighbors for k-nearest neighbors
R=1; % radius
P=8; % no. of points on the circle
%Processing the training data
TrainDataX=[];
TrainDataY=[];
%Looping over all the building images
files=dir('imagesDatabaseHW8/training/building/*.jpg');
img=cell(20,1);
i=1;
for file=files'
    img{i}=imread(strcat('imagesDatabaseHW8/training/building/',file.name));
    i=i+1;
end
parfor i=1:20
    f=LocalBP(img{i},R,P);
    TrainDataX=[TrainDataX;f'];
    TrainDataY=[TrainDataY;1];
end


%Looping over all the car images
files=dir('imagesDatabaseHW8/training/car/*.jpg');
img=cell(20,1);
i=1;
for file=files'
    img{i}=imread(strcat('imagesDatabaseHW8/training/car/',file.name));
    i=i+1;
```

```
end
parfor i=1:20
    f=LocalBP(img{i},R,P);
    TrainDataX=[TrainDataX;f'];
    TrainDataY=[TrainDataY;2];
end


%Looping over all the mountain images
files=dir('imagesDatabaseHW8/training/mountain/*.jpg');
img=cell(20,1);
i=1;
for file=files'
    img{i}=imread(strcat('imagesDatabaseHW8/training/mountain/',file.name));
    i=i+1;
end
parfor i=1:20
    f=LocalBP(img{i},R,P);
    TrainDataX=[TrainDataX;f'];
    TrainDataY=[TrainDataY;3];
end

%Looping over all the tree images
files=dir('imagesDatabaseHW8/training/tree/*.jpg');
img=cell(20,1);
i=1;
for file=files'
    img{i}=imread(strcat('imagesDatabaseHW8/training/tree/',file.name));
    i=i+1;
end
parfor i=1:20
    f=LocalBP(img{i},R,P);
    TrainDataX=[TrainDataX;f'];
    TrainDataY=[TrainDataY;4];
end


%Processing the testing data
TestDataX=[];
TestDataY=[];
%Looping over all the building images
files=dir('imagesDatabaseHW8/testing/building*.jpg');
img=cell(5,1);
i=1;
for file=files'
    img{i}=imread(file.name);
    i=i+1;
end
parfor i=1:5
    f=LocalBP(img{i},R,P);
    TestDataX=[TestDataX;f'];
    TestDataY=[TestDataY;1];
```

```
end


%Looping over all the car images
files=dir('imagesDatabaseHW8/testing/car*.jpg');
img=cell(5,1);
i=1;
for file=files'
    img{i}=imread(file.name);
    i=i+1;
end
parfor i=1:5
    f=LocalBP(img{i},R,P);
    TestDataX=[TestDataX;f'];
    TestDataY=[TestDataY;2];
end

%Looping over all the mountain images
files=dir('imagesDatabaseHW8/testing/mountain*.jpg');
img=cell(5,1);
i=1;
for file=files'
    img{i}=imread(file.name);
    i=i+1;
end
parfor i=1:5
    f=LocalBP(img{i},R,P);
    TestDataX=[TestDataX;f'];
    TestDataY=[TestDataY;3];
end


%Looping over all the tree images
files=dir('imagesDatabaseHW8/testing/tree*.jpg');
img=cell(5,1);
i=1;
for file=files'
    img{i}=imread(file.name);
    i=i+1;
end
parfor i=1:5
    f=LocalBP(img{i},R,P);
    TestDataX=[TestDataX;f'];
    TestDataY=[TestDataY;4];
end


% Snippet for plotting the histogram

% For building
figure;
bar(TrainDataX(1,:))
```

```
title('Building LBP Histogram','FontWeight','bold')

% For car
figure;
bar(TrainDataX(21,:))
title('Car LBP Histogram','FontWeight','bold')

% For mountain
figure;
bar(TrainDataX(41,:))
title('Mountain LBP Histogram','FontWeight','bold')

% For tree
figure;
bar(TrainDataX(61,:))
title('Tree LBP Histogram','FontWeight','bold')

% Snippet for plotting histogram ends

%Training a K-Nearest neighbour
mdl=fitcknn(TrainDataX, TrainDataY, 'NumNeighbors',nneighbors, 'distance', 'euclidean', ...
'Standardize',1);

%Testing a K-Nearest Neighbor
TestDataPred=mdl.predict(TestDataX);

%Confusion Matrix is plotted
Target=zeros(4,size(TestDataX,2));
Pred=zeros(4,size(TestDataX,2));

for i=1:size(TestDataX)
    Target(TestDataY(i),i)=1;
    Pred(TestDataPred(i),i)=1;
end
figure;
plotconfusion(Pred,Target);
xlabel('Output Class','FontWeight','bold')
ylabel('Target Class','FontWeight','bold')
```