Assignment 4

Face Detection
Overview

• large number of initial weak classifiers
• each weak classifier computes one rectangular feature
• the program computes the best threshold and polarity for each weak classifier
• Adaboost selects a subset of these classifiers and assigns a weight to each one
• Final classifications of boxes in test images are based on a combination of the selected ones.
Initialize features

- **Given in the code base**
- Initializes all weak classifiers
- Chooses the upper left corner \((x,y)\) and the height and width \(h\) and \(w\) randomly (but from 0 to 1)
- Chooses type of box
  - vertical 2-box
  - horizontal 2-box
  - vertical 3-box
- Sets area
ComputeTrainingSetFeatures

• Given in the code base as a shell
• Calls two methods that you code
  – IntegralImage: computes the integral image for each training patch
  – ComputeFeatures: uses the integral image for each training patch to compute features for that patch, one for each weak classifier, and puts them in an array called features.
Training Data and Features

Weights | training patches | integral images | features array
--- | --- | --- | ---

w0  
w1  
w2  
w3  
w4  
w5

holds feature values for each (feature/patch) combo.
AdaBoost

• **Given in the code base**
• Starts with uniform weights on training patches
• For each weak classifier
  – sorts the feature values in ascending order
  – results of sort go in `featureSort` and `featureSortIdx`
  – selects `numWeakClassifiers` weak classifiers through calling `FindBestClassifier` for all candidates and selecting the ones with lowest errors
• updates weights on patches in `dataWeights`
• computes current total error for the training data and scores for each sample for debug purposes
Initializing features

Function ComputeTrainingSetFeatures
for(i=0;i<numTrainingExamples;i++)
{
    .....  
    ComputeFeatures(integralImage, 0, 0, patchSize,
    &(features[i*numWeakClassifiers]), weakClassifiers, numWeakClassifiers,
    patchSize);
}

feature offset1: i * numWeakClassifiers

Function ComputeFeatures
for(i=0;i<numWeakClassifiers;i++)
{
    ..... 
    features[i] += weakClassifiers[i].m_BoxSign[j]*
    sum/((double) (size*size));
}

feature offset2: offset1 + i

features iterates over classifiers first,
and then training examples.
Feature Sorting

features

Differents classifiers

features for 1st training example

2nd training example

last training example

Differents classifiers

features for 1st classifier

2nd classifier

last classifier

featureTranspose

features

Differents classifiers

Differents examples

Differents examples

Differents examples

featureSort (sorted by value)

featureSortIdx

in sorted order of training sample

j

j is the index of the ith largest feature in sorted order.

featureSort is only for ONE classifier at a time.
**findBestClassifier**

- **you write it**
- It is called by AdaBoost with a candidate classifier
- It is given the sort index which indexes into
  - features
  - weights
  - training labels
- Use it to go through the training samples *(in sorted order)*, compute error for the classifier using the formula from the lecture (slides 30-32).
- Return threshold, classifier weight, and polarity
Using the Sort Index: Example

<table>
<thead>
<tr>
<th>samples</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>labels</td>
<td>F</td>
<td>B</td>
<td>F</td>
<td>B</td>
<td>B</td>
</tr>
<tr>
<td>features</td>
<td>6</td>
<td>3</td>
<td>10</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>weights</td>
<td>1/5</td>
<td>1/5</td>
<td>1/5</td>
<td>1/5</td>
<td>1/5</td>
</tr>
</tbody>
</table>

The feature values are for one particular feature (classifier).

The index tells you the sorted order of the features.
Setting the Polarity

\[ \text{error} = \min (BG + (AFS - FS), FS + (ABG - BG)) \]

- When left < right, set polarity to 0
- Else set polarity to 1
# Threshold and Polarity Example

\[
error = \min (BG + (AFS - FS), FS + (ABG - BG))
\]

<table>
<thead>
<tr>
<th>samples</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>labels</td>
<td>F</td>
<td>B</td>
<td>F</td>
<td>B</td>
<td>B</td>
</tr>
<tr>
<td>features</td>
<td>6</td>
<td>3</td>
<td>10</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>weight</td>
<td>1/5</td>
<td>1/5</td>
<td>1/5</td>
<td>1/5</td>
<td>1/5</td>
</tr>
<tr>
<td>index</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

**initialize**
- AFS = 0
- ABG = 0
- besterr = 999999

AFS becomes sum of face sample weights = 2/5; ABG = 3/5

**step 0: idx = 4; FS stays 0; BG = 1/5**
error = \( \min(1/5 + (2/5-0), 0 + (3/5-1/5)) = 2/5 \)
besterr = 2/5; bestpolarity = 1; bestthreshold=1

**step 1: idx = 3; FS stays 0; BG = 2/5**
error = \( \min(2/5 + (2/5-0), 0 + (3/5-2/5)) = 1/5 \)
besterr = 1/5; bestpolarity = 1; bestthreshold=2
Threshold and Polarity Example

error = min (BG + (AFS - FS), FS + (ABG - BG))

samples                  0  1  2  3  4
labels                   F  B  F  B  B
features                 6  3 10  2  1
weight                   1/5 1/5 1/5 1/5 1/5
index                    4  3  1  0  2

initialize
AFS = 0
ABG = 0
besterr = 999999

step 2: idx = 1; FS stays 0; BG = 3/5
error = min(3/5 + (2/5-0), 0 + (3/5-3/5)) = 0
besterr = 0; bestpolarity = 1; bestthreshold = 3

step 3: idx = 0; FS = 1/5; BG = 3/5
error = min(3/5 + (2/5-1/5), 1/5 + (3/5-3/5)) = 1/5
NO CHANGE

step 4: idx = 2; FS = 2/5; BG = 3/5
error = min(3/5 + (2/5-2/5), 2/5 + (3/5-3/5)) = 2/5
NO CHANGE

RESULT

| 1 | 2 | 3 | 6 | 10 |

θ > 3