Pattern recognition systems – Lab 10

The AdaBoost method

1. Objectives

In this lab session we will study an ensemble classifier obtained using a method called AdaBoost (Adaptive Boosting). We will apply it for a binary classification problem on 2D points.

2. Theoretical Background

AdaBoost, short for Adaptive Boosting, is a machine learning meta-algorithm formulated by Yoav Freund and Robert Schapire in [1], who won the 2003 Gödel Prize for their work [2]. In this session the goal will be to separate 2D points into two classes, the class membership is given by the color of the points.

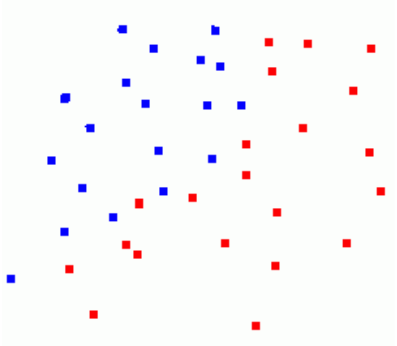

Figure 1 Example of two class samples (red and blue points)

The general idea of the AdaBoost algorithm is to build a strong classifier $H_T(\boldsymbol{x})$ which is the sign of the linear combination of T weak classifiers (or weak learners) h_t:

$$H_T(\boldsymbol{x}) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x})\right)$$

Each weak learner returns either +1 or -1 and is weighted by α_t. The final class is given by the sign of the strong classifier $H_T(\boldsymbol{x})$. In this work we will use **decision stumps** as weak learners. A decision stump classifies an instance by looking at a particular feature, if this feature is below a threshold, the instance is classified as class +1 and -1 otherwise.

We are given the training set in the following form: $\boldsymbol{X}$ is the feature matrix of dimension $n\,x\,m$ and contains n the training samples, each row being an individual sample of dimension m. In our case, $m = 2$ and the features are the rows and columns at which the points are found in the input image. The class vector $\boldsymbol{y}$ of dimension n contains +1 for each red point and -1 for each blue point.

For this method we will associate a weight with each example. We will store the weights in the weight vector $\boldsymbol{w}$ of dimension n. Initially all samples have an equal weight of $1/n$. The following algorithm describes the high level AdaBoost procedure which finds the strong classifier H_T.

Algorithm AdaBoost

```
init w_i=1/n
for t=1:T
  //also returns the weighted training error ε_t:
  [h_t, ε_t] = findWeakLearner(X,y,w)
  α_t = 0.5 ln((1−ε_t)/ε_t)
  s = 0
  for i=1:n
    //wrongly classified examples will have y_i h_t(X_i) < 0
    //their weights will be higher in the next step
    w_i ← w_i · exp(−α_t y_i h_t(X_i))
    s += w_i
  endfor
  //normalize the weights
  for i=1:n
    w_i ← w_i/s
  endfor
endfor
//returns all the alpha values and the weak learners
return [α, h]

[h_t, ε_t] = findWeakLearner(X,y,w)
best_h = {}
best_err = ∞
for j=1:X.cols
  for threshold=0:img.size //cols or rows
    for class_label={-1,1}
      e=0
      for i=1:X.rows
        if X(i,j)<threshold
          z_i=class_label
        else
          z_i=-class_label
        endif
        if z_i y_i < 0
          e += w_i
        endif
      endfor
      if e<best_err
        best_err = e
        best_h = {j, threshold, class_label, e}
      endif
    endfor
  endfor
endfor
return best_h
```

The underlying idea behind this algorithm is to find the best simple (weak) classifier and then to modify the importance of the examples. Missclassified examples will get a higher weight and correctly classified examples will get a lower weight. An example is classified as the wrong class if the sign of the expression $y_i h_t(X_i)$ is negative (the class labels have different signs).

At the following step, when we search for the next weak learner, it will be more important to correctly classify the examples which have high weights since they contribute more to the weighted training error.

Each weak learner contributes to the final score of the classifier. The contribution is weighted by how well the weak learner performed in terms of the weighted training error.

3. Implementation details

Suggested structure for a single weak leaner:

```
struct weaklearner{
  int feature_i;
  int threshold;
  int class_label;
  float error;
  int classify(Mat X){
    if (X.at<float>(feature_i)<threshold)
      return class_label;
    else
      return -class_label;
  }
};
```

Header for function that finds the best weak learner:

```
weaklearner findWeakLearner(Mat X, Mat y, Mat w)
```

Suggested structure for the strong classier (MAXT is a constant):

```
struct classifier{
  int T;
  float alphas[MAXT];
  weaklearner hs[MAXT];
  int classify(Mat X){
    return 0;
  }
};
```

Header for function which draws the decision boundary (keep the original image unmodified):

```
void drawBoundary(Mat img, classifier clf)
```

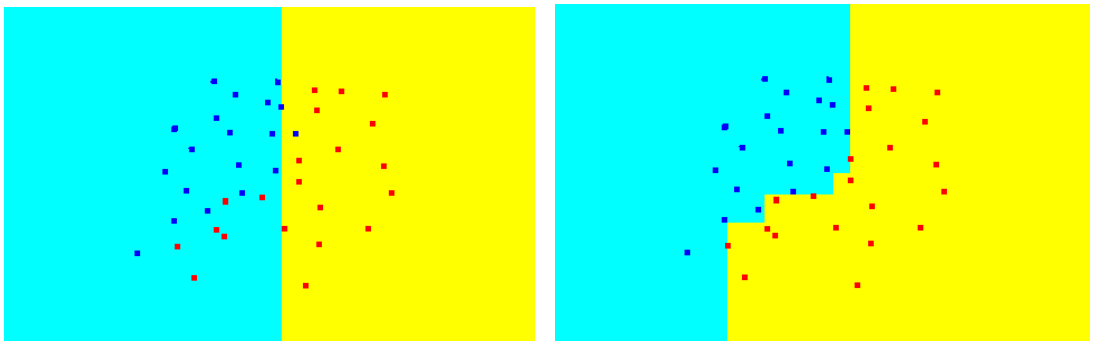

Figure 2. Sample results on points1 with T=1 (left) and T=13 (right)

4. Practical work

1. Read the training set from input files (points*.bmp). Each row from the feature matrix $\boldsymbol{X}$ should contain the row and column of each colored point from the image. The class vector $\boldsymbol{y}$ contains +1 for red and -1 for blue points.
2. Implement the decision stump weak learner.
3. Implement the `findWeakLearner` function.
4. Implement the `drawBoundary` function which colors the input image showing the decision boundary by changing the background color (white pixels) based on the classification result. Use yellow for +1 background and teal for -1 background pixels. Test the function with a strong classifier formed by a single weak learner.
5. Implement the AdaBoost algorithm to find the strong classifier with T weak learners. Visualize the decision boundary. For each input image find the value of T which results in zero classification error. What are the limitations of the presented method?

5. References

[1] Robert E. Schapire, The Boosting Approach to Machine Learning, An Overview, 2001

[2] AdaBoost - https://en.wikipedia.org/wiki/AdaBoost