Intelligent Systems: Reasoning and Recognition

James L. Crowley

ENSIMAG 2 and MoSIG M1 Winter Semester 2016

Lesson 4 12 February 2016

Boosted Learning with AdaBoost

Outline

Class notes on the web :

http://www-prima.inrialpes.fr/Prima/Homepages/jlc/Courses/2015/ENSI2.SIRR/ENSI2.SIRR.html

Notation

x_d	A feature. An observed or measured value.
$\vec{X}$	A vector of D features.
D	The number of dimensions for the vector $\vec{X}$
y	A dependent variable to be estimated.
$\{\vec{X}_m\}$ $\{y_m\}$	Training samples for learning.
M	The number of training samples.
$g_n(\vec{X})$	A discriminant function

$h_n(\vec{X}) = \text{sgn}(g_n(\vec{X})) = \begin{cases} 1 & \text{if } g_n(\vec{X}) \geq 0 \\ -1 & \text{if } g_n(\vec{X}) < 0 \end{cases}$ a decision (or hypothesis) function

$F_N(\vec{X}) = \sum_{n=1}^{N} \alpha_n h_n(\vec{X})$ A weighted Committee of weak classifiers

$I(z) = \begin{cases} 1 & z < 0 \\ 0 & z \geq 0 \end{cases}$ An error function for counting errors

AdaBoost

AdaBoost (adaptive Boosting) is a meta-algorithm for learning a 2-class detection function.

Adaboost builds a strong classifier from a large number of weak classifiers. The outputs of the weak classifiers are combined as a weighted sum of votes. The resulting strong committee can be made arbitrarily good by adding more weak classifiers.

Adaboost is particularly useful in problems with a large number of features or large numbers of possible weak classifiers.

The Boosted Classifier

The boosted classifier can be seen as a form of Committee that decides using weighted votes.

A weighted committee has the form:

$$F_N(\vec{X}) = \sum_{n=1}^{N} \alpha_n h_n(\vec{X})$$

where α_n is a learned weight for each decision function.
and $h_s(\vec{X})$ is a hypothesis or decision function.

The decision function uses N discriminant functions $g_n(\vec{X})$

$$h_n(\vec{X}) = \text{sgn}(g_n(\vec{X})) = \begin{cases} 1 & \text{if } g_n(\vec{X}) \geq 0 \\ -1 & \text{if } g_n(\vec{X}) < 0 \end{cases}$$

Each decision function $h_n(\vec{X})$ maps the output if a discrimiant $g_n(\vec{X})$ into a vote $v_n \in \{-1,1\}$

Boosted learning will choose the most effective discriminant functions from a much larger set $\{g_s(\vec{X})\}$ of possible discriminant functions.

Discriminant Functions

Adaboost is often used with linear discriminant functions, $g_s(\vec{X}) = \vec{w}_s^T \vec{X}$

where $\vec{X} = \begin{pmatrix} 1 \\ x_1 \\ \vdots \\ x_D \end{pmatrix}$ and $\vec{w} = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_D \end{pmatrix}$ as we saw in our previous lessons.

These are convenient because it is easy to compute a large number of possible linear functions from a set of training points.

However the algorithm can be used with any form of weak classifier.

It can even be used with individual features.

Let $\vec{\delta}_d = \begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix}$. If D is large, then we can use the D features $g_d(\vec{X}) = \vec{\delta}_d^T \vec{X}$

un this case the $g_d(\vec{X}) = \vec{X}_d + w_0$ which is a form of linear discriminant.

Boosted Learning

Boosting is an iterative algorithm that iteratively selects and adds the best decision functions to a weighted committee.

Assume training data composed of *M* sample observations $\{\vec{X}_m\}$ where each sample observation is labeled with an indicator variable $\{y_m\}$

$y_m = +1$ for examples of the target pattern (class 1)
$y_m = -1$ for all other examples (class 2)

The algorithm estimates a weight for each training sample, $b_m^{(i)}$. The weights are initially set to 1.

The Algorithm

1) Initialisation.

$$i \leftarrow 0;\ n \leftarrow 1; \quad \forall_{m=1}^{M} b_m = 1; \quad \alpha_1 \leftarrow 1;$$

$$\forall h_s() \in \{h_s()\}:\ E_s\left(h_s(\vec{X}_m)\right) = \sum_{m=1}^{M} I\left(h_s(\vec{X}_m)\right)$$

$$h_1(\vec{X}) = \arg\min_{h_s(\vec{X})} \left\{ \sum_{m=1}^{M} I\left(h_s(\vec{X}_m)\right) \right\}$$

where $I\left(h_s(\vec{X}_m)\right)$ is an error function that allows us to count the number of errors made by a decision function with the training data.

$$I\left(h_s(\vec{X}_m)\right) = \begin{cases} 1 & h_s(\vec{X}_m) < 0 \\ 0 & h_s(\vec{X}_m) \geq 0 \end{cases}$$

2) Loop until $E_i <$ Specified maximum Error rate.

$$i \leftarrow i+1$$

$$h_i(\vec{X}) = \arg\min_{h_s(\vec{X})} \left\{ \sum_{m=1}^{M} b_m \cdot I\left(y_m \cdot \left(h_s(\vec{X}_m) + \sum_{n=1}^{i-1} \alpha_n h_n(\vec{X}_m) \right) \right) \right\}$$

$$E_i = \frac{\sum_{m=1}^{M} b_m \cdot I\left(y_m \cdot \left(h_i(\vec{X}_m) + \sum_{n=1}^{i-1} \alpha_n h_n(\vec{X}_m)\right)\right)}{\sum_{m=1}^{M} b_m}$$

$$\alpha_i = \log\left(\frac{1-E_i}{E_i}\right)$$

$$\forall_{m=1}^{M}: \; b_m^{(i-1)} \cdot e^{\alpha_i I(y_m \cdot F_i(\vec{X}_m))}$$

end Loop. set $N \leftarrow i$; $\{\alpha_i\}$, $\{h_i(-)\}$

Explanation for each step

Initialisation.
a) initialize the index, i, the weights for training data b_m and for the first classifier α_1

$$n \leftarrow 1; \qquad \forall_{m=1}^{M} b_m = 1; \qquad \alpha_1 \leftarrow 1;$$

b) Choose the decision function, $h_s()$, that gives the lowest error rate with the training data .

$$h_1(\vec{X}) = \arg\min_{h_s(\vec{X})} \left\{ \sum_{m=1}^{M} I\left(h_s(\vec{X}_m)\right) \right\}$$

where $I\left(h_s(\vec{X}_m)\right)$ is an error function that allows us to count the number of errors made by each decision function

$$I\left(h_s(\vec{X}_m)\right) = \begin{cases} 1 & h_s(\vec{X}_m) < 0 \\ 0 & h_s(\vec{X}_m) \geq 0 \end{cases}$$

2) Loop until $E_i <$ Specified maximum Error rate.

a) Update the index variable $i \leftarrow i+1$

b) select the decision function that gives the lowest number of errors when added to the committee. assign this to the i^{th} decision function

$$h_i(\vec{X}) = \arg\min_{h_s(\vec{X})} \left\{ \sum_{m=1}^{M} b_m \cdot I\left(y_m \cdot \left(h_s(\vec{X}_m) + \sum_{n=1}^{i-1} \alpha_n h_n(\vec{X}_m) \right) \right) \right\}$$

c) compute the error rate for the updated committee

$$E_i = \frac{\sum_{m=1}^{M} b_m \cdot I\left(y_m \cdot \left(h_i(\vec{X}_m) + \sum_{n=1}^{i-1} \alpha_n h_n(\vec{X}_m) \right) \right)}{\sum_{m=1}^{M} b_m}$$

d) Compute the weight for the i^{th} decision function from the error rate

$$\alpha_i = \log\left(\frac{1 - E_i}{E_i} \right)$$

e) Update the weight for each training sample that is currently misclassified.

$$\forall_{m=1}^{M}: \; b_m^{(i-1)} \cdot e^{\alpha_i I(y_m \cdot F_i(\vec{X}_m))}$$

Note that when a training sample is correctly classifier, $I\left(y_m \cdot F_i(\vec{X}_m)\right) = 0$

end Loop.

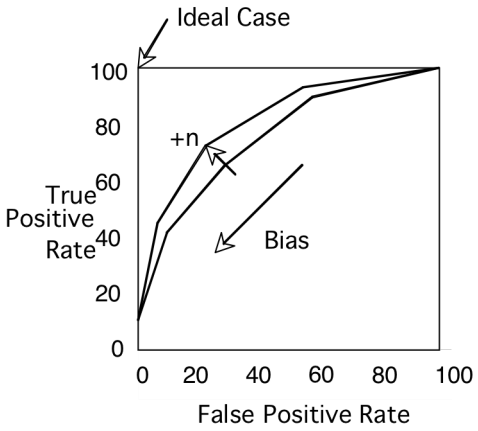
The algorithm ends when the error rate goes below a specified threshold.
It can also be ended after N loops, or when the TP rate falls below a threshold.

g) return i ; $\{\alpha_i\}$, $\{h_i(-)\}$

The committee is composed of the $N = i$ decision functions, with their weights.

ROC Curve for a weighted committee

As we saw Wednesday, the ROC plots the True Positive Rate (TPR) against False Positive Rate (FPR) for a classifier as a function of the global bias B.



The Boosting theorem states that adding a new boosted detectors to a committee always improves the committee's ROC curve. We can continue adding classifiers until we obtain a desired rate of false positives and false negatives.

However, in general, the improvement provided for each new classifier becomes progressively smaller. We can end up with a very very large number of classifiers.

The halting criteria for boosted learning is set in terms of the FPR and TPR. When the ROC curve goes above for point (FPR, TPR) for some Bias B, the algorithm halts.

Note that the probability of error for a committee of classifiers can be computed as

$$P(Error) = \frac{F}{N} = \frac{\#FP + \#FN}{N}$$

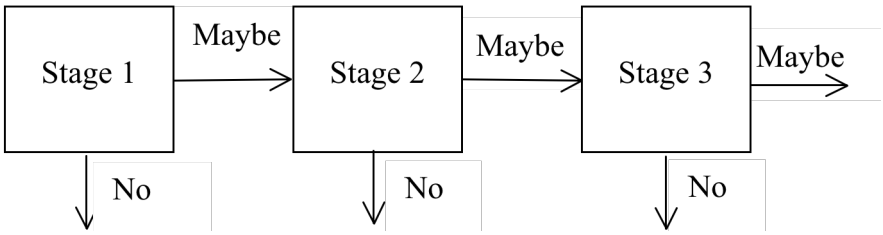
Where N is the number of training samples, and F is the number of False detections within the N training samples.

Learning a multi-stage cascade of classifiers

We can optimize the computation time by separating the committee into a multi-stage cascade of committees.

Each stage is composed of a committee that is designed with avoids rejecting possible true positives (high TPR: True Positive Rate) at the cost of accepting many False Positives (high False Positive Rate).

We construct each stage using only training data that passed the previous stage. Later stages are more expensive but are used less often.



For each stage we set a minimum acceptable target for True Positives using the training data and accept the false positive rate that results.

Note that this can result in over-fitting the training data. It is important that the training data represent as large a variety of data as possible.