GI01/4C55: Supervised Learning

8. Boosting

November 28, 2005

Massimiliano Pontil

Today's plan

- Weak learners
- Adaboost
- Properties of boosting

Bibliography: These lecture notes are available at:
http://www.cs.ucl.ac.uk/staff/M.Pontil/courses/index-SL05.htm
Lecture notes are based on Hastie, Tibshirani & Friedman, Chapter 10,1–5 and the tutorial by Rob Schapire: "The boosting approach to machine learning: an overview"

Spam email classification problem

Consider the problem of classifying an email you receive as "good email" or "spam email"

- Different features can be used to represent an email
  e.g. the bag of words representation (we may also use additional features to code the text in the subject section of the email etc.)
- Different learning methods (Naive Bayes, SVM, k-NN, CART, etc.) can be used to approximate this task...
- *Key observation:* it is easy to find "*rules of thumb*" that are often correct (say 60% of the time) e.g. "IF 'business' occurs in email THEN predict as spam"
- Hard to find highly accurate prediction rules

Weak learners and boosting

Weak learner: an algorithm which can consistently find classifiers ("rules of thumb") at least slightly better than random guessing, say better than 55%

Boosting: a general method of converting rough rules of thumb into a highly accurate prediction rule (classifier)

Boosting algorithm

- Devise a computer program for deriving rough rules of thumb
- Choose rules of thumb to fit a subset of example
- Repeat T times
- Combine the classifiers by weighted majority vote

key steps are:

- how do we choose the subset of examples at each round? concentrate on **hardest examples** (those most often misclassified by previous classifiers)
- how do we combine the weak learners? by **weighted majority**

Adaboost algorithm

Given training data $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$
Initialize $D_1(i) = \frac{1}{m}$

For $t = 1, \ldots, T$:

- fit a classifier $h_t : \mathbb{R}^d \to \mathbb{R}$ using distribution D_t
- choose $\alpha_t \in \mathbb{R}$
- update

$$D_{t+1}(i) = \frac{D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)}}{Z_t} \qquad (*)$$

  where Z_t is a normalization factor (so as to ensure that D_{t+1} is a distribution)

Output the final classifier $H(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right)$

Some remarks

- The basic idea in Adaboost is to maintain a distribution D on the training set and iteratively train a weak learner on it
- At each round larger weights are assigned to hard examples, hence the weak learner will focus mostly on those examples

Let ϵ_t be the weighted training error of classifier t:

$$\epsilon_t = \sum_{i=1}^{m} D_t(i) I\{h_t(\mathbf{x}) \neq y_i\}$$

We will justify later the following choice for α_t

$$\alpha_t = \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t} \qquad (1)$$

Weight by majority

$$\alpha_t = \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$$

The final classifier is a weighted majority vote of the T base classifiers where α_t is the weight assigned to h_t

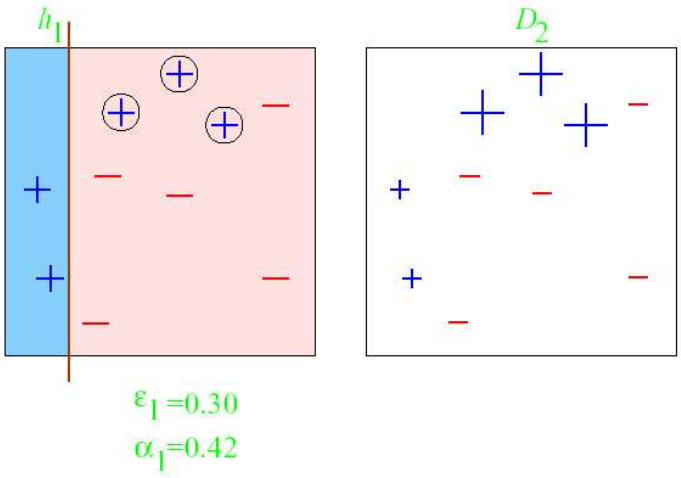
$$f(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t$$

Typically $\epsilon_t \leq 0.5$ hence $\alpha_t \geq 0$ (this is always the case if h_t and $-h_t$ are both in the set of weak learners, e.g. for decision stumps)

Thus, f is essentially a convex combination of the h_t with weights controlled by the training error

Example (round 1)

Let's discuss a simple example where the weak learners are decision stumps (vertical or horizontal lines, i.e. trees with only two leaves)
the 2nd plot highlights the difficult examples

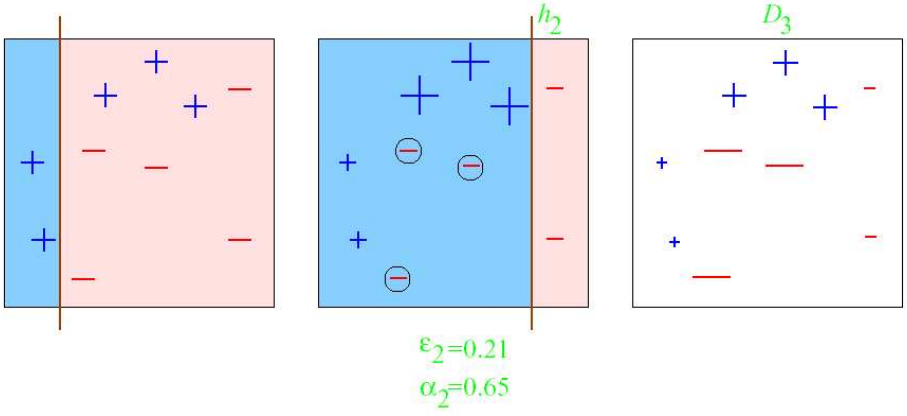


(Thanks to Rob Schapire for providing the figures for this example)

Example (round 2)

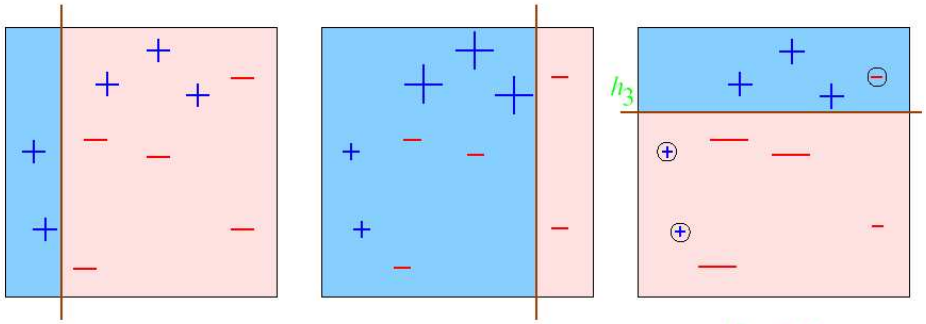
The second classifier concentrates more on the difficult examples

Examples are then re-weighted according to this classifier



Example (round 3)

In this example the set of weak learners can be assumed to be finite. There are roughly $4m$ weak learners/decision stumps corresponding to choosing either the horizontal or vertical coordinate and splitting between any two points

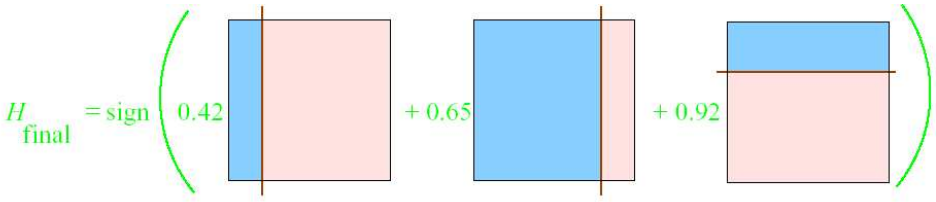


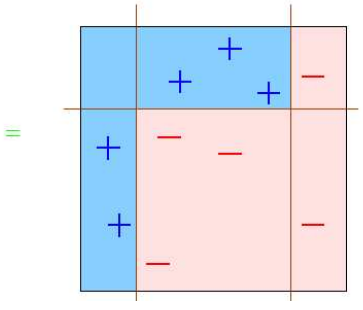
$\varepsilon_3 = 0.14$

$\alpha_3 = 0.92$

Example (final classifier)

The final classifier has zero training error!





Analysis of the training error (I)

The training error of the boosting algorithm is bounded as

$$\frac{1}{m}\sum_{i=1}^{m} I\{H(\mathbf{x}_i) \neq y_i\} \leq \frac{1}{m}\sum_{i=1}^{m} e^{-y_i f(\mathbf{x}_i)} = \prod_{t=1}^{T} Z_t$$

where we have defined $f = \sum_t \alpha_t h_t$ (so that $H(\mathbf{x}) = \text{sign}(f(\mathbf{x}))$)

The inequality follows from: $H(\mathbf{x}_i) \neq y_i \;\Rightarrow\; e^{-y_i f(\mathbf{x}_i)} \geq 1$

The equality follows from the recursive definition of D_t (see also page 19)

Analysis of the training error (II)

The previous bound suggests that if at each iteration we choose α_t and h_t by minimizing Z_t the final training error of H will be reduced most rapidly

Generally we choose $h_t : \mathbb{R}^d \to \mathbb{R}$ (margin classifiers). However, if we constrain h_t to have range $\{-1, 1\}$ (so these are binary classifiers), Z_t is minimized by the choice in equation (1) above

$$\alpha_t = \frac{1}{2}\log\frac{1-\epsilon_t}{\epsilon_t}$$

We show this next

Analysis of the training error (III)

From formula (*) on page 6 we have

$$Z_t = \sum_i D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)}$$

Using the fact that h_t returns binary values we also have that

$$\begin{aligned} Z_t &= e^{\alpha_t} \sum_{i:y_i \neq h_t(\mathbf{x}_i)} D_t(i) + e^{-\alpha_t} \sum_{i:y_i = h_t(\mathbf{x}_i)} D_t(i) \\ &= \epsilon_t e^{\alpha_t} + (1-\epsilon_t) e^{-\alpha_t} \end{aligned}$$

Equation (1) now easy follows by solving $\frac{dZ_t}{d\alpha_t} = 0$

Analysis of the training error (IV)

Placing $\alpha_t = \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$ in the above formula for Z_t we obtain

$$Z_t = \epsilon_t e^{\alpha_t} + (1-\epsilon_t) e^{-\alpha_t} = 2\sqrt{\epsilon_t(1-\epsilon_t)} = \sqrt{1-4\gamma_t^2}$$

where $\gamma_t = 1/2 - \epsilon_t$. Hence we have the following bound for the training error

$$Error \leq \prod_t Z_t \leq \prod_t \sqrt{1-4\gamma_t^2} \leq e^{-2\sum_t \gamma_t^2}$$

Thus, if each weak classifier is slightly better than random guessing (if $\gamma_t \geq \gamma > 0$) the training error drops **exponentially fast**

$$Error \leq e^{-2T\gamma^2}$$

Additive models and boosting

Boosting can be seen as a greedy way to solve the problem

$$\min_{\mathbf{w}} \sum_{i=1}^{m} V(y_i, \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_i))$$

where the feature vector $\boldsymbol{\phi} = (\phi_1, \phi_2, \ldots, \phi_N)$ is formed only by weak learners

At each iteration a new basis function is added to the current basis expansion $f^{(t-1)} = \sum_{s=1}^{t-1} \alpha_s h_s$

$$(\alpha_t, n(t)) = \text{argmin}_{\alpha,n} \sum_{i=1}^{m} V(y_i, f^{(t-1)}(\mathbf{x}_i) + \alpha \phi_n(\mathbf{x}_i))$$

and $h_t = \phi_{n(t)}$. This is unlike in CART and MARS, where at each iteration previous basis function coefficients are re-adjusted

Additive models and boosting

$$\min_{\alpha_t, n} \sum_{i=1}^{m} V(y_i, f^{(t-1)}(\mathbf{x}_i) + \alpha_t \phi_n(\mathbf{x}_i))$$

In the statistical literature, this type of algorithm is called *forward stagewise additive model*

It is also possible to establish a relation between boosting and L^1 norm regularization

$$\min_{\mathbf{w}} \sum_{i=1}^{m} V(y_i, \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_i)) + \lambda \sum_{n=1}^{N} |w_n|$$

which says that essentially boosting maximizes the L^1 margin $(\sum_{n=1}^{N} |w_n|)^{-1}$

Why the exponential loss?

We will show when V is the exponential loss that

$$e^{-y_i(f^{(t-1)}(\mathbf{x}_i)+\alpha_t h_t(\mathbf{x}_i))} \propto D_t(i)e^{-y_i\alpha_t h_t(\mathbf{x}_i)}$$

In fact, using formula (*) on page 6 recursively it is easy to see that

$$D_t(i) = \frac{e^{-y_i f^{(t-1)}(\mathbf{x}_i)}}{m\prod_{s=1}^{t-1} Z_s}$$

from which, summing over i, it follows that

$$D_t(i) = \frac{e^{-y_i f^{(t-1)}(\mathbf{x}_i)}}{\sum_{j=1}^{m} e^{-y_j f^{(t-1)}(\mathbf{x}_j)}}$$

Note: as a special case we have that $\prod_{t=1}^{T} Z_t = \frac{1}{m}\sum_{i=1}^{m} e^{-y_i f(\mathbf{x}_i)}$ showing the equality on page 13

Why the exponential loss? (cont.)

The formula

$$e^{-y_i(f^{(t-1)}(\mathbf{x}_i)+\alpha_t h_t(\mathbf{x}_i))} \propto D_t(i)e^{-y_i\alpha_t h_t(\mathbf{x}_i)}$$

implies that

$$\begin{aligned}\sum_{i=1}^{m} e^{-y_i\left(f^{(t-1)}(\mathbf{x}_i)+\alpha_t h_t(\mathbf{x}_i)\right)} &\propto \sum_{i=1}^{m} D_t(i)e^{-y_i\alpha_t h_t(\mathbf{x}_i)} \\ &= (e^{\alpha_t} - e^{-\alpha_t})\epsilon_t(h_t) + e^{-\alpha_t}\end{aligned}$$

where as before $\epsilon_t(h_t) = \sum_{i=1}^{m} D_t(i)I\{h_t(\mathbf{x}_i) \neq y_i\}$

Hence, for a fixed value of $\alpha_t > 0$, minimizing the exponential loss w.r.t. h_t is the same as minimizing w.r.t. the weighted misclassification error!

Summarizing

In summary, Adaboost can be interpreted as a greedy way to minimize the exponential loss criterion via the forward-stagewise additive model approach

Alternatively, let $\{h_n\}_{n=1}^N$ the set of all weak learners (assume they are finite in number). Adaboost minimizes

$$\sum_{i=1}^{m} e^{-y_i \sum_{n=1}^{N} w_n \phi_n(\mathbf{x}_i)}$$

by coordinate descent, at each iteration t choosing coordinate $h_t = \phi_{n(t)}$ which produces the biggest decrease in error and updating $w_{n(t)} = \alpha_t$

Boosting and logistic regression

The expected error w.r.t. the exponential loss

$$\mathbf{E}_{\mathbf{x},y}\left[e^{-yf(\mathbf{x})}\right] = \mathbf{E}_{\mathbf{x}}\left[P(y=1|\mathbf{x})e^{-f(\mathbf{x})} + P(y=-1|\mathbf{x})e^{f(\mathbf{x})}\right]$$

is minimized (exercise) for

$$f^*(x) = \frac{1}{2}\log\frac{P(y=1|\mathbf{x})}{P(y=-1|\mathbf{x})} \quad \Rightarrow \quad P(y=1|\mathbf{x}) = \frac{1}{1+e^{-2f^*(\mathbf{x})}}$$

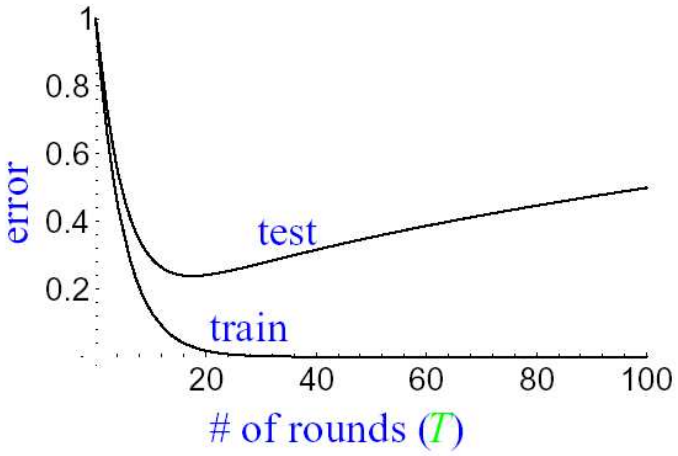
The last formula can be used to convert the output of Adaboost into a probability estimate

Generalization error (I)

How do we expect training and test error of boosting to depend on T?

At first sight, we'd expect that if T is too large over-fitting kicks off

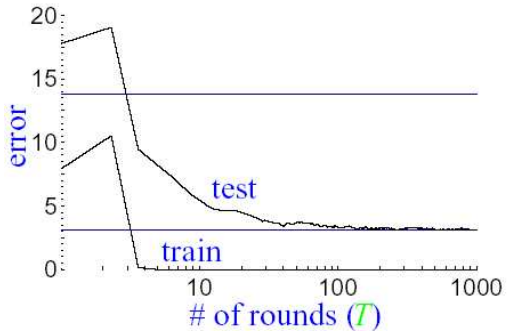
However, the plot shown here is not typically the case...

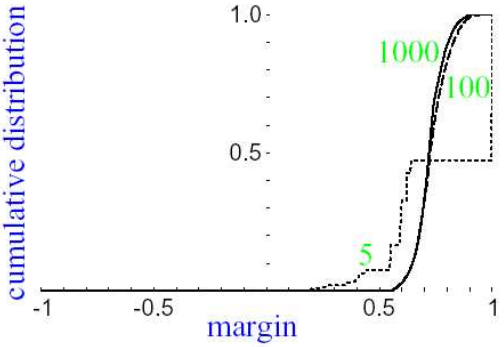


(Figures from Schapire *et. al*)

Generalization error (II)

Typically, the test error keeps decreasing even when the training error is zero! (eventually it will increase but may take many more iterations to do so)





However the **margin distribution** keeps decreasing as well which explains why test error does so

Generalization error (III)

The margin of point i is simply the quantity $\text{margin}_i = y_i f(\mathbf{x}_i)$ where we assume that $\sum_t \alpha_{t=1}^T = 1$ (just normalize the weights after the last iteration of boosting).

The margin distribution is the empirical distribution of the margin

One can show that with high probability (say 99%)

$$\text{generalization error} \leq \Pr_{\text{emp}}(\text{margin} \leq \theta) + O\left(\frac{\sqrt{d/m}}{\theta}\right)$$

where m is the number of training points, d the VC-dimension of the set of weak learners ($\log N$ in our case) and $\Pr_{\text{emp}}$ denotes empirical probability of the margin (like the training error this converges exponentially fast to zero)