

## Boosting and ensemble methods

Ensemble methods are used to form combinations of classifiers whose accuracy is better than that of each single classifier in the combination. Fix a training set  $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)$  for a binary classification problem with zero-one loss, and assume classifiers  $h_1, \dots, h_T$  are available such that

$$\mathbb{P}(h_1(\mathbf{x}_Z) \neq y_Z \wedge \dots \wedge h_T(\mathbf{x}_Z) \neq y_Z) = \prod_{i=1}^T \mathbb{P}(h_i(\mathbf{x}_Z) \neq y_Z) \quad (1)$$

where  $Z$  is a random variable uniformly distributed on the set  $\{1, \dots, m\}$  of indices of training examples. In other words, we assume that the error regions of each classifier be independent with respect to the uniform distribution over the training set. These error regions define the training error  $\widehat{\ell}(h_i)$  of each classifier  $h_i$ . Indeed,

$$\widehat{\ell}(h_i) = \frac{1}{m} \sum_{t=1}^m \mathbb{I}\{h_i(\mathbf{x}_t) \neq y_t\} = \mathbb{P}(h_i(\mathbf{x}_Z) \neq y_Z)$$

Since we use the zero-one loss, we may also assume that  $\widehat{\ell}(h_i) \leq \frac{1}{2}$  for each  $i = 1, \dots, T$ . Indeed, if  $\widehat{\ell}(h_i) > \frac{1}{2}$  for some  $h_i$ , then the binary classifier  $-h_i$  satisfies  $\widehat{\ell}(h_i) < \frac{1}{2}$ .

Now consider the majority classifier  $f$  defined by

$$f(\mathbf{x}) = \operatorname{sgn} \left( \sum_{i=1}^T h_i(\mathbf{x}) \right)$$

The majority classifier is a simple example of an *ensemble method*, combining the predictions of an ensemble of classifiers in order to boost the accuracy. Note that  $f(\mathbf{x}) = 1$  if and only if at least half of the classifiers  $h_1, \dots, h_T$  predict 1 on  $\mathbf{x}$  (assume  $T$  is odd to avoid ties). We now bound the training error of  $f$ ,

$$\begin{aligned} \widehat{\ell}(f) &= \mathbb{P}(f(\mathbf{x}_Z) \neq y_Z) \\ &= \mathbb{P} \left( \sum_{i=1}^T \mathbb{I}\{h_i(\mathbf{x}_Z) \neq y_Z\} > \frac{T}{2} \right) \\ &= \mathbb{P} \left( \frac{1}{T} \sum_{i=1}^T \mathbb{I}\{h_i(\mathbf{x}_Z) \neq y_Z\} > \ell_{\text{ave}} + \left( \frac{1}{2} - \ell_{\text{ave}} \right) \right) \end{aligned}$$

where we defined

$$\ell_{\text{ave}} = \frac{1}{T} \sum_{i=1}^T \widehat{\ell}(h_i)$$

Now introduce Bernoulli random variables  $G_1, \dots, G_m$  defined by  $G_i = \mathbb{I}\{h_i(\mathbf{x}_Z) \neq y_Z\}$ . Note that these random variables are independent, due to our assumption (1). Also,  $\mathbb{E}[G_i] = \ell(h_i)$  and

$$\frac{1}{T} \sum_{i=1}^T \mathbb{E}[G_i] = \ell_{\text{ave}} < \frac{1}{2}$$

A slight generalization of Chernoff-Hoeffding bounds to independent random variables with unequal expectations gives

$$\mathbb{P}\left(\frac{1}{T} \sum_{i=1}^T G_i > \ell_{\text{ave}} + \varepsilon\right) \leq e^{-2\varepsilon^2 T}$$

This immediately gives  $\widehat{\ell}(f) \leq \exp\left(-2T\left(\frac{1}{2} - \ell_{\text{ave}}\right)^2\right)$  that, by letting  $\gamma_i = \frac{1}{2} - \widehat{\ell}(h_i)$  we may also write as

$$\widehat{\ell}(f) \leq \exp\left(-2 \sum_{i=1}^T \gamma_i^2\right) \quad (2)$$

This result tells us that if we manage to obtain classifiers with independent training errors in the sense of (1), then the training error of the majority vote classifier decreases exponentially with respect to  $\gamma_1, \dots, \gamma_T$ , where each  $\gamma_i$  measures how better is a classifier  $h_i$  on the training set when compared to random guessing.

But how can we obtain classifiers with independent training errors? A popular heuristic, known as Bagging, applies to any learning algorithm  $A$  for binary classification and to any training set  $S$ . Let  $m$  be the size of  $S$ . Bagging builds  $h_1, \dots, h_T$  by drawing  $m$  examples uniformly at random with replacement from  $S$ . This process is repeated  $T$  times so to obtain the resampled training sets  $S_1, \dots, S_T$ . Then  $A$  is run on each  $S_i$  setting  $h_i = A(S_i)$ . The idea is that the resampling procedure helps enforce condition (1).

One may wonder how different from  $S$  any  $S_i$  can be. To find that out, we take a little detour and compute the fraction of unique data points in  $S_i$ . As you see in a moment, more than one third of the points of  $S$  are missing from  $S_i$  in expectation! Let  $N$  be the number of unique points drawn, and let  $X_t$  be the indicator function of the event that  $(\mathbf{x}_t, y_t)$  is drawn. Then the probability that  $(\mathbf{x}_t, y_t)$  is not drawn is

$$\mathbb{P}(X_t = 0) = \left(1 - \frac{1}{m}\right)^m$$

So we have

$$\mathbb{E}[N] = \sum_{t=1}^m \mathbb{E}[X_t] = \sum_{t=1}^m \mathbb{P}(X_t = 1) = \sum_{t=1}^m \left(1 - \left(1 - \frac{1}{m}\right)^m\right) = m - m \left(1 - \frac{1}{m}\right)^m$$

Therefore, the fraction of unique points in  $S_i$  is

$$1 - \left(1 - \frac{1}{m}\right)^m \approx 1 - \frac{1}{e} = 0.632\dots$$

where the approximation becomes exact for  $m \rightarrow \infty$ .

We now introduce **boosting**, a principled ensemble method that achieves the exponential bound (2) on the training error without requiring the demanding condition (1). Boosting is an incremental method to build classifiers of the form  $\text{sgn}(f)$ , where

$$f = \sum_{i=1}^T w_i h_i$$

and  $\mathbf{w} = (w_1, \dots, w_T)$  is a vector of real coefficients. We assume  $h_1, \dots, h_T$  belong to some family  $\mathcal{H}$  of **base classifiers**.

In practice, also in order to save computational costs, base classifiers are very simple. A typical choice for  $\mathcal{H}$  is that of **decision stumps**. These are all classifiers of the form  $h_{i,\tau} : \mathbb{R}^d \rightarrow \{-1, 1\}$  defined by  $h_{i,\tau}(\mathbf{x}) = \text{sgn}(x_i - \tau)$ , where  $i = 1, \dots, d$  and  $\tau \in \mathbb{R}$ .

The specific boosting algorithm we introduce is known as **AdaBoost** (adaptive boosting). Fix a training set  $S$  with  $m$  examples  $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)$ , and a sequence  $h_1, \dots, h_T$  of base classifiers. We now show how to choose the coefficients  $\mathbf{w}$  so that the training error is bounded as in (2).

AdaBoost uses the convex upper bound  $\mathbb{I}\{z \leq 0\} \leq e^{-z}$  on the zero-one loss function  $\mathbb{I}\{f(\mathbf{x}_t) y_t < 0\}$ . This gives

$$\widehat{\ell}_S(f) = \frac{1}{m} \sum_{t=1}^m \mathbb{I}\{f(\mathbf{x}_t) y_t < 0\} \leq \frac{1}{m} \sum_{t=1}^m e^{-f(\mathbf{x}_t) y_t} = \frac{1}{m} \sum_{t=1}^m e^{-\sum_{i=1}^T w_i h_i(\mathbf{x}_t) y_t}$$

Note that other algorithms use different convex upper bounds on the zero-one loss. For example, SVM uses the hinge loss.

Introduce now the functions  $L_1, \dots, L_T$  defined by  $L_i(t) = h_i(\mathbf{x}_t) y_t$ . Note that  $L_i(t) \in \{-1, 1\}$  and  $L_i(t) = 1$  if and only if  $h_i(\mathbf{x}_t) = y_t$ . Recalling that  $Z$  is a random variable uniformly distributed in  $\{1, \dots, m\}$ , we can view each  $L_i(Z)$  as a random variable and write

$$\widehat{\ell}_S(f) \leq \frac{1}{m} \sum_{t=1}^m e^{-\sum_{i=1}^T w_i L_i(t)} = \mathbb{E} \left[ e^{-\sum_{i=1}^T w_i L_i} \right] = \mathbb{E} \left[ \prod_{i=1}^T e^{-w_i L_i} \right]$$

If condition (1) were true, we could write the expectation of the product as a product of expectations. In order to sidestep the condition, we change the probability space and write

$$\mathbb{E} \left[ \prod_{i=1}^T e^{-w_i L_i} \right] = \prod_{i=1}^T \mathbb{E}_i \left[ e^{-w_i L_i} \right] \tag{3}$$

where each expectation  $\mathbb{E}_i [e^{-w_i L_i}]$  is understood with respect to a probability  $\mathbb{P}_i$  yet to be specified.

Assuming (3) holds, which we verify later, we can proceed as follows

$$\begin{aligned}
\widehat{\ell}_S(f) &\leq \prod_{i=1}^T \mathbb{E}_i [e^{-w_i L_i}] \\
&= \prod_{i=1}^T (e^{-w_i} \mathbb{P}_i(L_i = 1) + e^{w_i} \mathbb{P}_i(L_i = -1)) \\
&= \prod_{i=1}^T (e^{-w_i} (1 - \varepsilon_i) + e^{w_i} \varepsilon_i)
\end{aligned} \tag{4}$$

where we set

$$\varepsilon_i \stackrel{\text{def}}{=} \mathbb{P}_i(L_i = -1) = \sum_{t=1}^m \mathbb{I}\{L_i(t) = -1\} \mathbb{P}_i(t)$$

Note that  $\varepsilon_i$  is the error of  $h_i$  with respect to the probability  $\mathbb{P}_i$ . Namely,  $\varepsilon_i$  is the weighted training error of  $h_i$  where the weights are determined by  $\mathbb{P}_i$ .

Before computing the  $\mathbb{P}_i$ , we show how to pick  $w_1, \dots, w_T$  in order to minimize (4). By computing the zeros of the derivative of  $e^{-w}(1 - \varepsilon_i) + e^w \varepsilon_i$  with respect to  $w$ , we find a single zero at

$$w = \frac{1}{2} \ln \frac{1 - \varepsilon_i}{\varepsilon_i}.$$

Note that the above expression is only defined for  $0 < \varepsilon_i < 1$ . As we will see,  $\mathbb{P}(t) > 0$  for all  $t \in \{1, \dots, m\}$ . Hence  $\varepsilon_i \in \{0, 1\}$  implies that either  $h_i$  or  $-h_i$  has zero training error on  $S$ . If this happens, then we can throw away all  $h_j$  for  $j \neq i$  and avoid using boosting altogether. Therefore, without loss of generality we may assume  $0 < \varepsilon_i < 1$  for all  $i = 1, \dots, m$ .

Through the study of the derivative, we find that the unique zero corresponds to a minimum of the function  $e^{-w}(1 - \varepsilon_i) + e^w \varepsilon_i$ . Hence we set

$$w_i \stackrel{\text{def}}{=} \frac{1}{2} \ln \frac{1 - \varepsilon_i}{\varepsilon_i} \quad \text{for } i = 1, \dots, T.$$

Substituting in (4) and simplifying, we get

$$\widehat{\ell}_S(f) \leq \prod_{i=1}^T \sqrt{4\varepsilon_i(1 - \varepsilon_i)}.$$

Note that  $w_i = 0$  if and only if  $\varepsilon_i = 1/2$ , meaning that the weight (according to  $\mathbb{P}_i$ ) of the training points where  $h_i$  errs is exactly  $\frac{1}{2}$ . Because such a  $h_i$  does not affect the value of  $f$  (for the very reason that  $\varepsilon_i = 0$ ) without loss of generality we may also assume that  $\varepsilon_i \neq 1/2$ .

Set  $\gamma_i \stackrel{\text{def}}{=} 1/2 - \varepsilon_i$  and note that  $\varepsilon_i \neq 1/2$  implies  $\gamma_i \neq 0$ . Using the inequality  $1 + x \leq e^x$ , which holds for all  $x \in \mathbb{R}$ , we get

$$\widehat{\ell}_S(f) \leq \prod_{i=1}^T \sqrt{4\varepsilon_i(1 - \varepsilon_i)} = \prod_{i=1}^T \sqrt{1 - 4\gamma_i^2} \leq \prod_{i=1}^T e^{-2\gamma_i^2} = e^{-2 \sum_{i=1}^T \gamma_i^2}$$

This is the same bound as the one we proved in (2) under the condition (1). Note, however, that the definition of  $\gamma_i = \frac{1}{2} - \varepsilon_i$  changes because  $\varepsilon_i$  now is the weighted training error of  $h_i$ .

The bound  $\widehat{\ell}_S(f) \leq e^{-2\sum_{i=1}^T \gamma_i^2}$  is pretty powerful. For example, if we had  $|\gamma_i| > \gamma > 0$  for each  $i = 1, \dots, T$  then  $\widehat{\ell}_S(f) \leq e^{-2T\gamma^2}$ . In other words, the training error goes down exponentially fast in the number  $T$  of boosting rounds. In this case, using the observation that  $\widehat{\ell}_S(f) = 0$  if and only if  $\widehat{\ell}_S(f) < 1/m$ , we conclude that a number

$$T > \frac{\ln m}{2\gamma^2}$$

of boosting rounds is sufficient to bring the training error of  $f$  down to zero.

We now move on to derive the  $\mathbb{P}_1, \dots, \mathbb{P}_T$  satisfying condition (3). Setting  $\mathbb{P}_1 = \mathbb{P}$ ,  $\mathbb{E}_1 = \mathbb{E}$ , and

$$\mathbb{P}_{i+1}(t) = \frac{\mathbb{P}_i(t)e^{-w_i L_i(t)}}{\mathbb{E}_i[e^{-w_i L_i}]} \quad \text{for } t = 1, \dots, m \text{ and } i = 1, \dots, T-1 \quad (5)$$

where

$$\mathbb{E}_i[e^{-w_i L_i}] = \sum_{s=1}^m e^{-w_i L_i(s)} \mathbb{P}_i(s)$$

It is easy to check that  $\mathbb{P}_1, \dots, \mathbb{P}_T$  are indeed probability distributions on  $\{1, \dots, m\}$ . In particular,  $\mathbb{P}_i(t) > 0$  and  $\mathbb{P}_i(1) + \dots + \mathbb{P}_i(m) = 1$ .

Per questa scelta delle distribuzioni di probabilità possiamo ora dimostrare la come segue. Prima di tutto risolviamo la ottenendo For this choice of  $\mathbb{P}_i$  we can prove (3) as follows. First, we solve (5) for  $e^{-w_i L_i(t)}$  obtaining

$$e^{-w_i L_i(t)} = \mathbb{E}_i[e^{-w_i L_i}] \frac{\mathbb{P}_{i+1}(t)}{\mathbb{P}_i(t)}$$

Then, we write

$$\begin{aligned} \mathbb{E} \left[ \prod_{i=1}^T e^{-w_i L_i} \right] &= \frac{1}{m} \sum_{t=1}^m \left( \prod_{i=1}^T \mathbb{E}_i[e^{-w_i L_i}] \frac{\mathbb{P}_{i+1}(t)}{\mathbb{P}_i(t)} \right) \\ &= \frac{1}{m} \sum_{t=1}^m \left( \frac{\mathbb{P}_{T+1}(t)}{\mathbb{P}_1(t)} \right) \left( \prod_{i=1}^T \mathbb{E}_i[e^{-w_i L_i}] \right) \\ &= \left( \sum_{t=1}^m \frac{\mathbb{P}_{T+1}(t)}{\mathbb{P}_1(t)} \mathbb{P}_1(t) \right) \left( \prod_{i=1}^T \mathbb{E}_i[e^{-w_i L_i}] \right) \\ &= \prod_{i=1}^T \mathbb{E}_i[e^{-w_i L_i}] \end{aligned}$$

concluding the proof.

These probability distributions have a simple interpretation when one studies how  $\mathbb{P}_{i+1}$  depends on  $\mathbb{P}_i$ . Fix  $\mathbb{P}_i$  and suppose  $\varepsilon_i < 1/2$ . Then  $w_i > 0$  and each  $\mathbb{P}_{i+1}(t)$  is obtained multiplying  $\mathbb{P}_i(t)$  for the quantity  $e^{-w_i L_i(t)}$ , which is bigger than one if and only if  $h_i(\mathbf{x}_t) \neq y_t$ . In other words, the

weight of each training example  $(\mathbf{x}_t, y_t)$  is increased when  $\mathbb{P}_i$  is updated to  $\mathbb{P}_{i+1}$  if and only if  $h_i$  errs on  $(\mathbf{x}_t, y_t)$ . Intuitively, the boosting process concentrates the weight on the training examples that are misclassified by the previous classifiers. A similar argument applies to the case when  $\varepsilon_i > 1/2$ .

We are now ready to introduce the pseudo-code for AdaBoost. The only thing we left unspecified is how the sequence  $h_1, \dots, h_T$  of classifiers is generated. In principle, we want a sequence such that the corresponding sequence  $\gamma_1, \dots, \gamma_T$  is as far away from  $\frac{1}{2}$  as possible (in either direction) so to ensure an exponential decrease of the training error. It is convenient to view the boosting process as sequence of rounds between the boosting algorithm and a generic learning algorithm  $A$ . In each round  $i$ , the booster gives  $\mathbb{P}_i$  to  $A$  and gets  $h_i$  in response. The interaction goes on for  $T$  rounds, unless  $A$  return  $h_i$  such that  $\gamma_i = 0$ . In that case the boosting process stops and the booster outputs  $f = \text{sgn}(w_1 h_1 + \dots + w_{i-1} h_{i-1})$ .

Input:     Training set  $S$  of examples  $(\mathbf{x}, y) \in \mathbb{R}^d \times \{-1, 1\}$ .  
            Learning algorithm  $A$ .  
            Maximum number  $T$  of boosting rounds.

Initialize  $\mathbb{P}_1(t) \leftarrow 1/m$  for  $t = 1, \dots, m$ .

For  $i = 1, \dots, T$

1. Feed  $A$  with  $S$  weighted by  $\mathbb{P}_i$  and get  $h_i$  in response
2. Compute  $\varepsilon_i$  for  $h_i$
3. If  $\varepsilon_i \in \{0, \frac{1}{2}, 1\}$  then BREAK
4. Let  $w_i \leftarrow \frac{1}{2} \ln \frac{1-\varepsilon_i}{\varepsilon_i}$ .
5. Compute  $\mathbb{P}_{i+1}$  using (5).

If for loop exited on BREAK, then deal with the special case  
 Else output  $f = \text{sgn}(w_1 h_1 + \dots + w_T h_T)$ .

Additional topics that are left to cover: risk bounds for boosting and consistency of boosting.