Introduction to Machine Learning **Fall Semester**

Recitation 10

Lecturer: Regev Schweiger *Scribe: Yishay Mansour and RS*

10.1 AdaBoost - Review

Reminder: Algorithm A learns Weak-PAC a concept class C with H if:
$\exists \gamma > 0$,
$\forall c^* \in C$, (target function)
$\forall D$, (distribution)
$\forall \delta < \frac{1}{2}$,
With probability $1-\delta$, algorithm A outputs an hypothesis $h \in H$ such that $error(h) \leq \frac{1}{2} - \gamma$.

Input: A set of m classified examples: $S = \{\langle \mathbf{x}_1, y_1 \rangle, \langle \mathbf{x}_2, y_2 \rangle, \cdots, \langle \mathbf{x}_m, y_m \rangle\}$ where $\mathbf{x}_i \in \mathbb{R}^d, y_i \in \{-1, 1\}$ and a class of weak learners H.
Definitions: Let D_t denote the distribution (weights) of the examples at iteration t: $D_t(x_i)$ is the weight of example $\langle \mathbf{x}_i, y_i \rangle$ at iteration t.
Initialization:

$$D_1(\mathbf{x}_i) = \frac{1}{m} \quad \forall i \in \{1, \cdots, m\}$$

Iterate: $t = 1, 2, \ldots, T$

$$\begin{aligned} h_t &= \arg\min_{h \in H} \Pr_{\mathbf{x} \sim D_t} [h(\mathbf{x}) \neq y] \\ \epsilon_t &= \Pr_{\mathbf{x} \sim D_t} [h_t(\mathbf{x}) \neq y] \\ \alpha_t &= \frac{1}{2} \ln \frac{1 - \epsilon_t}{\epsilon_t} \\ D_{t+1}(\mathbf{x}_i) &= \frac{D_t(\mathbf{x}_i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)}}{Z_t} \end{aligned}$$

Finally:

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

Figure 10.1 illustrates the AdaBoost algorithm.

[1] Adapted from *Foundations of Machine Learning*

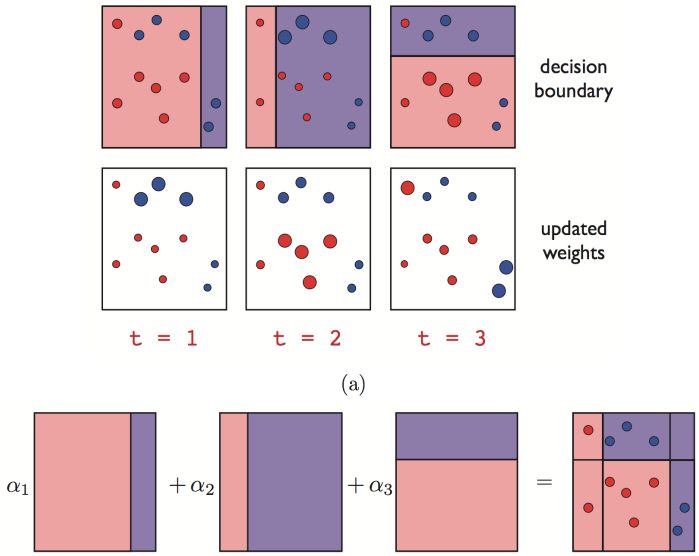

Figure 10.1: Example of AdaBoost with axis-aligned hyperplanes as base learners. (a) The top row shows decision boundaries at each boosting round. The bottom row shows how weights are updated at each round, with incorrectly (resp., correctly) points given increased (resp., decreased) weights. The size of the points represents the distribution weight assigned to them at each boosting round. (b) Visualization of final classifier, constructed as a linear combination of base learners.[1]

10.2 Coordinate Descent and AdaBoost

AdaBoost was designed to address a novel theoretical question, that of designing a strong learning algorithm using a weak learning algorithm. We will show, however, that it coincides in fact with a very simple and classical algorithm, which consists of applying a coordinate descent technique to a convex and differentiable objective function. Coordinate descent is just like gradient descent, except that you can't move along the gradient, you have to choose just one coordinate at a time to move along.

Suppose (for the sake of simplicity) that there is a finite number of weak classifiers, $h_j(\mathbf{x})$, $j = 1, \ldots, n$. We wish to find a linear combination with coefficients λ_j, that minimizes the average exponential loss:

$$F(\boldsymbol{\lambda}) = F(\lambda_1, \ldots, \lambda_n) = \frac{1}{m} \sum_{i=1}^{m} e^{-y_i \sum_{j=1}^{n} \lambda_j h_j(\mathbf{x}_i)}$$

The coordinate descent algorithm will first choose a coordinate k in which the slope is

the steepest. This translates to calculating, for each k, the following:

$$\begin{aligned}\frac{\partial}{\partial\alpha}F(\boldsymbol{\lambda}+\alpha\cdot\mathbf{e}_k) &= \frac{1}{m}\sum_{i=1}^{m}\frac{\partial}{\partial\alpha}\left(e^{-y_i\sum_{j=1}^{n}(\lambda_j+\alpha\cdot\delta_{j=k})h_j(\mathbf{x}_i)}\right)\\ &= -\frac{1}{m}\sum_{i=1}^{m}y_ih_k(\mathbf{x}_i)e^{-y_i\sum_{j=1}^{n}(\lambda_j+\alpha\cdot\delta_{j=k})h_j(\mathbf{x}_i)}\end{aligned}$$

Define $D(\mathbf{x}_i)=e^{-y_i\sum_{j=1}^{n}\lambda_jh_j(\mathbf{x}_i)}/Z$, with Z the relevant normalizing factor. Define ϵ_k as the error of the k-th classifier, according to D: $\epsilon_k=\Pr_{\mathbf{x}\sim D}(h_k(\mathbf{x})\neq y)$. Substituting $\alpha=0$ gives:

$$\begin{aligned}&= -\frac{1}{m}\sum_{i=1}^{m}y_ih_k(\mathbf{x}_i)e^{-y_i\sum_{j=1}^{n}\lambda_jh_j(\mathbf{x}_i)}\\ &= -\frac{1}{m}\sum_{i=1}^{m}y_ih_k(\mathbf{x}_i)D(\mathbf{x}_i)Z\\ &\propto \sum_{i=1}^{m}y_ih_k(\mathbf{x}_i)D(\mathbf{x}_i)\\ &= (1-\epsilon_k)-(\epsilon_k)=1-2\epsilon_k\end{aligned}$$

So finding the direction k the maximizes the steepness (absolute value) of the slope is equivalent to minimizing ϵ_k, which is what AdaBoost does at each step!

The next stage in this coordinate descent variant is to analytically calculate the optimal step α is direction k. The derivative can be further expanded:

$$\begin{aligned}\frac{\partial}{\partial\alpha}F(\boldsymbol{\lambda}+\alpha\cdot\mathbf{e}_k) &= -\frac{1}{m}\sum_{i=1}^{m}y_ih_k(\mathbf{x}_i)e^{-y_i\sum_{j=1}^{n}(\lambda_j+\alpha\cdot\delta_{j=k})h_j(\mathbf{x}_i)}\\ &= -\frac{1}{m}\sum_{i=1}^{m}y_ih_k(\mathbf{x}_i)e^{-y_i\sum_{j=1}^{n}\lambda_jh_j(\mathbf{x}_i)}\cdot e^{-y_i\sum_{j=1}^{n}\alpha\cdot\delta_{j=k}h_j(\mathbf{x}_i)}\\ &= -\frac{1}{m}\sum_{i=1}^{m}y_ih_k(\mathbf{x}_i)D(\mathbf{x}_i)Z\cdot e^{-y_i\alpha\cdot h_k(\mathbf{x}_i)}\end{aligned}$$

The last expression, $e^{-y_i\alpha\cdot h_k(\mathbf{x}_i)}$ can take two values, e^{α} and $e^{-\alpha}$, according to if $y_i=h_k(\mathbf{x}_i)$.

We can group these terms to get

$$\begin{aligned}
&= -\frac{1}{m}\left(\sum_{\{i|y_i=h_k(\mathbf{x}_i)\}} y_i h_k(\mathbf{x}_i) D(\mathbf{x}_i) Z \cdot e^{-\alpha} + \sum_{\{i|y_i\neq h_k(\mathbf{x}_i)\}} y_i h_k(\mathbf{x}_i) D(\mathbf{x}_i) Z \cdot e^{\alpha}\right) \\
&= -\frac{Z}{m}\left(e^{-\alpha}\sum_{\{i|y_i=h_k(\mathbf{x}_i)\}} D(\mathbf{x}_i) - e^{\alpha}\sum_{\{i|y_i\neq h_k(\mathbf{x}_i)\}} D(\mathbf{x}_i)\right) \\
&= -\frac{Z}{m}\left(e^{-\alpha}(1-\epsilon) - e^{\alpha}\epsilon\right) = 0
\end{aligned}$$

Equating to 0 and solving gives

$$\alpha = \frac{1}{2}\log\left(\frac{1-\epsilon}{\epsilon}\right)$$

So, the optimal step size is the weight in AdaBoost. Finally, updating D is AdaBoost is the same as updating D according to our current λ-s.