AdaBoost (Freund & Schapire '97)

"AdaBoost" is short for "Adaptive Boosting"

- very widely used, one of the most popular machine learning algorithms
- "An Empirical Comparison of Supervised Learning Algorithms" Caruana & Niculescu Mizil '06
  1) Boosted DT
  ⋮
  SVM
  ⋮
  Boosted Stumps
  ⋮
  DT
  ⋮
  Log Reg
- "Top 10 Algorithms in Data Mining" (Wu et al '08) presents algs identified by IEEE Conference ICDM in Dec 2006 (popularity contest)
  C4.5
  ⋮
  SVM
  ⋮
  AdaBoost
  CarT
- Freund & Schapire received the 2003 Gödel Prize for their work on AdaBoost.
  - "meta" algorithm - can be used to improve ("boost") the performance of another learning algorithm
  - very easy to implement, there are no standard software packages because you just do it yourself.

A Little History:

AdaBoost came out of the PAC Learning community.

- PAC Learning Model was developed by Valiant 1984
- Kearns & Valiant (1988, 1994) posed the question of whether a "weak" PAC learning alg, e.g. one that performs slightly better than a random guess, could be used to construct a "strong" PAC Learning algorithm, i.e., one that performs arbitrarily well (yet still poly-time & other technical conditions.)
- Freund & Schapire's example: betting strategies for horse racing:
  An expert gambler may not be able to explain his/her betting strategy, but when presented w/ data for a specific set of races, s/he can give us a "rule of thumb"
  - bet on the horse that has won the most recently
  - bet on the horse w/ the most favored odds

  Rules of thumb are not very accurate, but a little better than a random guess.

  Boosting algorithms combine these rules of thumb into a single, highly accurate prediction rule.
- Schapire (1989). showed that the answer to Kearns & Valiant's question was "yes". Proof by construction of the 1st boosting algorithm (wasn't so practical though!)
- Between '89-'95 a few other boosting algs were designed but none were very practical
- Freund & Schapire (1995) "A decision-theoretic generalization of on-line learning & an application to boosting (1997) - introduced AdaBoost.
- Schapire & Singer (1999) extended AdaBoost to more general rules of thumb.

Let's derive AdaBoost. (But not in the way F&S did it. Turns out that AdaBoost is stagewise optimization, discovered by at least 5 different groups. We'll do it that way.)

Feb 19-2:35 PM

Standard Classification Task :

training set : $\{(x_i, y_i)\}_{i=1..m}$ $x \in \mathcal{X}$, $y \in \{-1, 1\}$
chosen randomly from unknown prob. dist'n.

Want to find $f: \mathcal{X} \to \mathbb{R}$ such that $\text{sign}(f(x))$ agrees with y as much as possible, $f \in \mathcal{F}$.

(We hope that if our chosen f performs well on the training set, it will perform well on the whole prob. distn.)

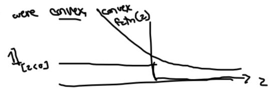
misclassification error of f w.r.t. training set $=$ # of times $y_i \neq \text{sign}(f(x_i))$

$=$ # of times $y_i f(x_i) < 0$

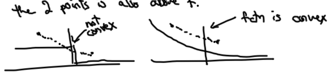
$$= \sum_{i=1}^{m} \mathbb{1}_{[y_i f(x_i) < 0]}$$

← indicator function is 1 if condition holds, 0 otherwise

We want misclassification error to be small, i.e. we would like to choose f to minimize misclassification error. The problem is that in terms of practical optimization, minimizing this quantity is very difficult. It would be much easier if the misclassification error were convex.



Aside: A function is convex if and only if the set of points lying on or above the graph is a convex set. Pick any 2 pts above the graph of f. If f is convex, the whole line of points connecting the 2 points is also above f.



For convex fctns, all local minima are global minima

Sums of convex functions are convex, and other nice properties.

Let's perform a trick, namely to use:

(*) $$\mathbb{1}_{[z<0]} \leq e^{-z}$$

Thus, misclassific error of f w.r.t. training set $= \sum_{i=1}^{m} \mathbb{1}_{[y_i f(x_i) < 0]}$

$$\leq \sum_{i=1}^{m} e^{-y_i f(x_i)} =: L(f)$$

We hope that choosing an f to yield small values of $L(f)$ will yield small values of the misclassification error.

Next, need to choose the form of f. For AdaBoost, f is a linear combination of "weak classifiers" or "rules of thumb".

$$f(x) = \sum_{j=1}^{n} \lambda_j h_j(x) \quad \text{where } h_j : \mathcal{X} \to \{-1, 1\}$$

AdaBoost's Objective Function : $$L(\lambda) = \sum_{i=1}^{m} e^{-\sum_{j=1}^{n} \lambda_j y_i h_j(x_i)}$$

Want to minimize this ↗ w.r.t. λ.

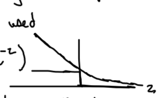
About the weak classifiers $\{h_j\}_{j=1...n}$

- AdaBoost can be used in 2 ways, either to do most of the work or as a "wrapper" to increase the accuracy of an already accurate base learning algorithm (e.g. boosted neural networks). In the second case, the h_j's are classifiers that come from the base learning algorithm.
- n can be huge, or even infinite.
- We assume that the h_j's are given to us, so we just need to find λ.

Aside: Derivation of Logistic Regression's Objective :

If, instead of (*) we had used

$$\mathbb{1}_{[z<0]} \leq \log(1 + e^{-z})$$



then we would derive the objective for the classification alg called logistic regression

$$L_{LogReg}(\lambda) = \sum_{i=1}^{m} \log\left(1 + e^{-\sum_{j=1}^{n} \lambda_j y_i h_j(x_i)}\right)$$

Aside: Part of SVM's Objective :

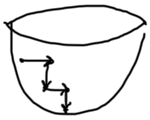
If, instead of (*) we had used

$$\mathbb{1}_{[z<0]} \leq \begin{cases} 1-z & z \leq 1 \\ 0 & z \geq 1 \end{cases}$$

then we would have derived part of the objective for SVM's.
(More another day.)

Back to AdaBoost:

Since $L(\lambda)$ is convex in λ, we can use simple techniques to minimize $L(\lambda)$ w.r.t λ in $\mathbb{R}^n$. We'll use "coordinate descent".



for $t=1\ldots T$

Step 1: Find the steepest "direction" j_t (i.e. choose a weak classifier)

Step 2: move along that direction until $L(\lambda)$ is minimized

(i.e. choose α to minimize $L(\lambda+\alpha e_{j_t})$ where $e_{j_t} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix}$ $\leftarrow j_t$

Formally,

Objective $L(\lambda) = \sum_{i=1}^m e^{-\sum_j \lambda_j y_i h_j(x_i)}$, $L(\lambda+\alpha e_{j_t}) = \sum_{i=1}^m e^{-\sum_j \lambda_j y_i h_j(x_i) - \alpha y_i h_{j_t}(x_i)}$

Directional derivative $\displaystyle -\frac{dL(\lambda+\alpha e_j)}{d\alpha}\Big|_{\alpha=0} = \sum_{i=1}^m + y_i h_j(x_i)\, e^{-\sum_j (\lambda_j y_i h_j(x_i) + \cancel{\alpha y_i h_j(x_i)})}\Big|_{\alpha=0}$

Step 1: $\displaystyle j_t = \operatorname{argmax}_j -\frac{dL(\lambda+\alpha e_j)}{d\alpha}\Big|_{\alpha=0} = \operatorname{argmax}_j \sum_i y_i h_j(x_i)\, \underbrace{e^{-\sum_j (\lambda_j y_i h_j(x_i))}}$

$\displaystyle = \operatorname{argmax}_j \sum_{i=1}^m y_i h_j(x_i)\, d_i$ where $\displaystyle d_i = \frac{e^{-\sum_j \lambda_j y_i h_j(x_i)}}{Z}$ $\leftarrow$ normalize.

Step 2: $\displaystyle 0 = -\frac{dL(\lambda+\alpha e_{j_t})}{d\alpha} \propto \sum_{i=1}^m + y_i h_{j_t}(x_i)\, \frac{e^{-\sum_j (\lambda_j y_i h_j(x_i)) + \alpha y_i h_{j_t}(x_i))}}{Z}$

$\displaystyle = \sum_{i:\, y_i h_{j_t}(x_i)=1} +1\, d_i e^{-\alpha} + \sum_{i:\, y_i h_{j_t}(x_i)=-1} -1\, d_i e^{\alpha}$

$= (1-d_-)e^{-\alpha} - d_- e^{\alpha}$ where $d_- = \sum_{i:\, y_i h_{j_t}(x_i)=-1} d_i$

$(1-d_-)e^{-\alpha} = d_- e^{\alpha}$

$\displaystyle \frac{1-d_-}{d_-} = e^{2\alpha} \longrightarrow \alpha = \frac{1}{2}\ln\frac{1-d_-}{d_-}$

Pseudocode for AdaBoost:

Given $\{(x_i,y_i)\}_{i=1\ldots m}$ $\{h_j\}_{j=1\ldots n}$, T, $\lambda_{1j}=0\ \forall j$

For $t=1\ldots T$, $d_{1,i} = 1/m\ \forall i$

$\displaystyle j_t = \operatorname{argmax}_j \sum_{i=1}^m y_i h_j(x_i)\, d_i$ "train weak learner using dist d"

$\displaystyle d_{t-} = \sum_{i:\, y_i h_{j_t}(x_i)=-1} d_{t,i}$ "error of weak classifier"

$\displaystyle \alpha_t = \frac{1}{2}\ln\left(\frac{1-d_{t-}}{d_{t-}}\right)$

$\lambda_{t+1} = \lambda_t + \alpha_t e_{j_t}$ add weak classifier's contribution

$d_{t+1,i} = d_{t,i} \cdot \left\{ \begin{array}{ll} e^{-\alpha_t} & \text{if } h_{j_t}(x_i) = y_i \\ e^{\alpha_t} & \text{if } h_{j_t}(x_i) \neq y_i \end{array} \right\} / Z_t$

write weights for next round in terms of weights for this round

end

To evaluate f, $f(x) = \sum_j \lambda_{T,j} h_j(x)$.

$\square$