Last time, we derived objectives that upper bd the misc. error

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

$$L(f) = \sum_{i=1}^{m} e^{-y_i f(x_i)} \quad \text{AdaBoost}$$

$$L_{\text{LogReg}}(f) = \sum_{i=1}^{m} \log\left(1 + e^{-y_i f(x_i)}\right) \quad \text{Logistic Regression}$$

$$L_{\text{part of SVM}}(f) = \sum_{i=1}^{m} \ell(y_i f(x_i)) \quad \text{where } \ell(z) = \begin{cases} 1-z & z \leq 1 \\ 0 & z \geq 1 \end{cases} \quad \text{part of SVM}$$

We derived AdaBoost, which minimizes $L(f)$, $f(x) = \sum_{j=1}^{n} \lambda_j h_j(x)$

Pseudocode:

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

for $t = 1 .. T$

$j_t = \underset{j}{\text{argmax}} \ y_i h_j(x_i) d_{t,i}$ "train weak learner"

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

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

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

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

end

Recall: $d_{t,i} = \dfrac{e^{-\sum_j \lambda_{t,j} y_i h_j(x_i)}}{Z_t}$, where $Z_t = \sum_{i=1}^{m} e^{-\sum_j \lambda_{t,j} y_i h_j(x_i)} = L(\lambda)$

Training Error Bound:

$L(\lambda)$ decreases at every iteration: $L(\lambda_{t+1}) = \sum_{i=1}^{m} e^{-\left(\sum_j y_i h_j(x_i) \lambda_{t+1,j}\right)}$ ↓ use def λ_{t+1}

$$L(\lambda_{t+1}) = \sum_{i=1}^{m} e^{-\left(\sum_j y_i h_j(x_i) \lambda_{tj}\right)} e^{-\alpha_t y_i h_{j_t}(x_i)}$$ ↓ use def of d_{ti}

$$= L(\lambda_t) \sum_i d_{ti} e^{-\alpha_t y_i h_{j_t}(x_i)}$$ ↓ use def of d_{t-}

$$= L(\lambda_t) \left[d_{t-} e^{\alpha_t} + (1-d_{t-}) e^{-\alpha_t} \right]$$ ↓ where $\alpha_t = \frac{1}{2} \ln\left(\frac{1-d_{t-}}{d_{t-}}\right)$

$$= L(\lambda_t) \left(d_{t-}(1-d_{t-})\right)^{1/2} 2$$ ↓ recurse

$$= L(\lambda_1) \prod_t 2 \left(d_{t-}(1-d_{t-})\right)^{1/2}$$

$e^{\alpha_t} = \left(\frac{1-d_{t-}}{d_{t-}}\right)^{1/2}$

Also, proportion of misclassification errors =

$$\frac{1}{m} \sum_{i=1}^{m} \cdot \mathbb{1}_{[y_i f_{\lambda_T}(x_i) < 0]} \leq \frac{1}{m} L(f_{\lambda_T})$$

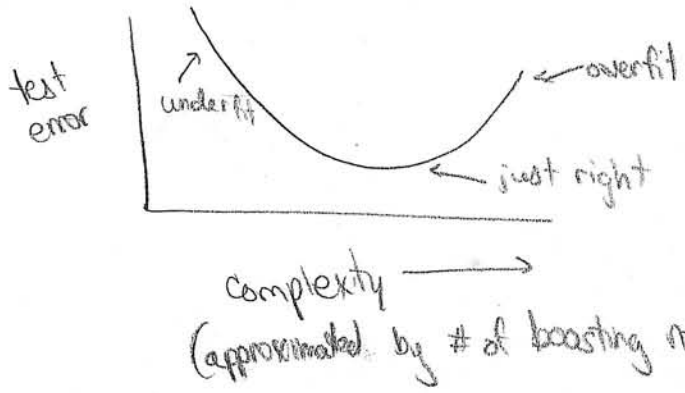
$$= \frac{1}{m} L(\lambda_1) \prod_t 2 \left(d_{t-}(1-d_{t-})\right)^{1/2}$$ } Training Error Bound

as long as d_{t-} is $< 1/2$, the error decreases at each iteration

Note: In fact, as long as d_{t-} is away from $\frac{1}{2}$, the error decreases. AdaBoost automatically moves along either h_{j_t} or $-h_{j_t}$ depending on d_{t-}. You can see this by considering $\alpha_t = \frac{1}{2} \ln\left(\frac{1-d_{t-}}{d_{t-}}\right)$ which can be either (+)ve or (-)ve.

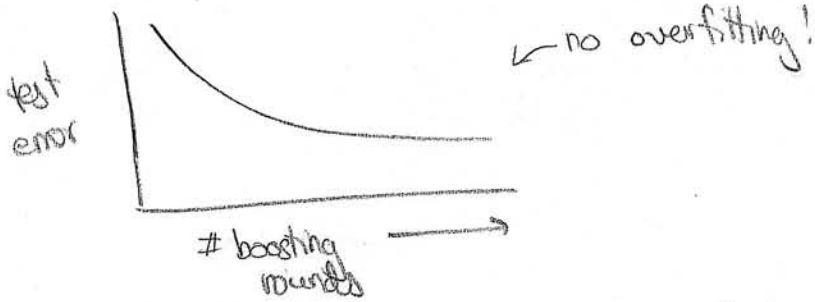
Something Unexpected: AdaBoost Tends Not to Overfit

The Bias/Variance Curve we Expect:



Designers of AdaBoost warned users against running AdaBoost for too many rounds to prevent overfitting. Luckily, experimentalists ignored this advice!

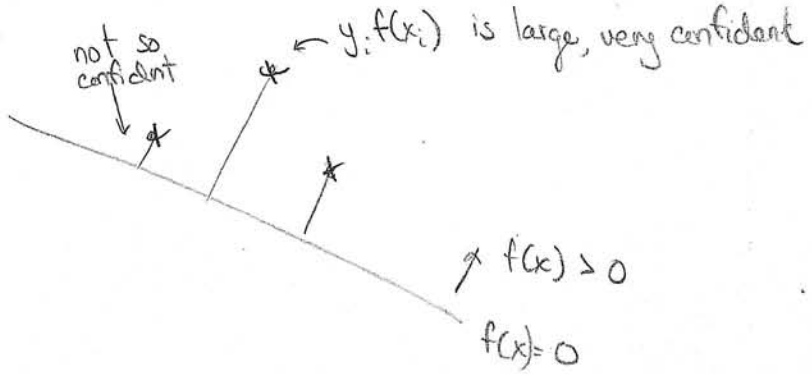
Drucker & Cortes '96, Quinlan '96, Breiman '98 found curves like this instead:



Seems to contradict Occam's Razor, ie. that less complex solutions are better.

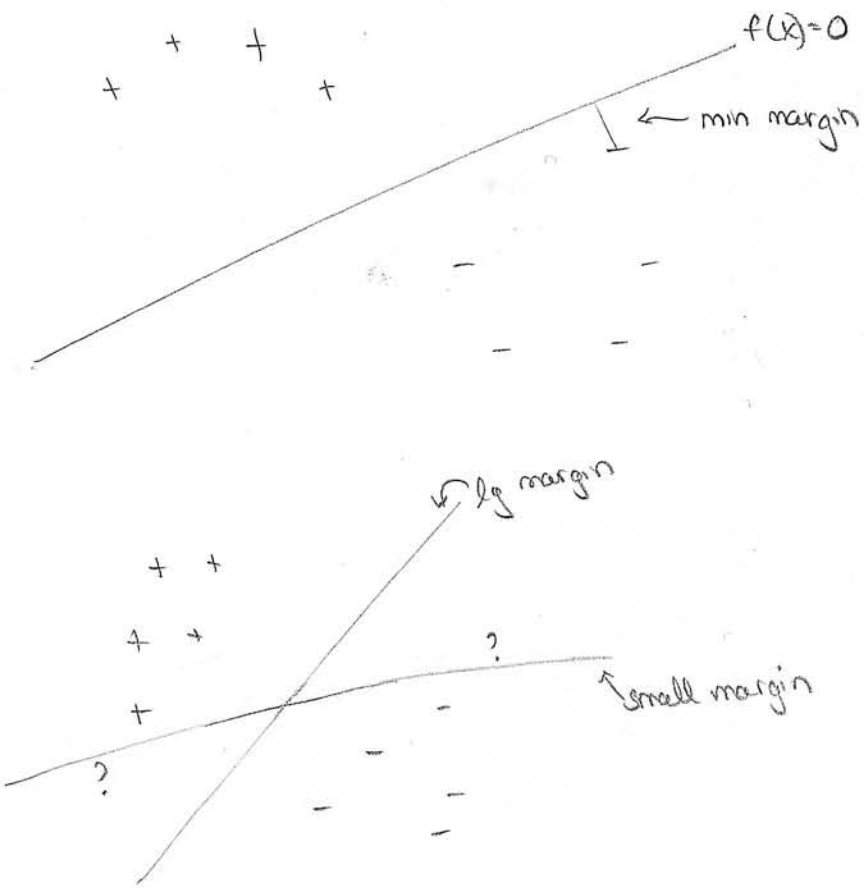
The experimental results show that AdaBoost works better than expected, but the designers wanted to explain why. Their explanation is in "Boosting the Margin" (Schapire, Freund, Bartlett, Lee '98). Here, complexity ≠ # of boosting rounds. The key is to use the margin. You'll see. the explanation is still not complete!

The margin of training example i is proportional to $y_i f(x_i)$, which is a confidence measure of a classifier's ability, & is like the "distance" from the training example to the decision boundary.



AdaBoost seems to be very good at achieving large margins.

The <u>minimum margin</u> (or the <u>margin</u> of the classifier f) is the min of the margins over training examples, i.e. the "distance" from the decision boundary to the nearest example:



Around the same time SVM's (Vapnik, Cortes & Vapnik '95) were becoming popular. SVM's are designed to maximize the margin (whereas AdaBoost was not!)

"Boosting the Margin" has 2 results:

1) Large margins are good (statistical guarantee)

Pseudotheorem: with high probability,

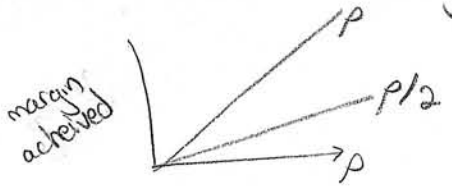
misclassification error on test set $\leq$ misclassification error on training set + function $\left((VCdim)^{1/2}, \frac{1}{\sqrt{m}}, \frac{1}{\text{margin}}\right)$

This is why it is believed that large margins are the key to success!

2) AdaBoost achieves large margins

Pseudotheorem:

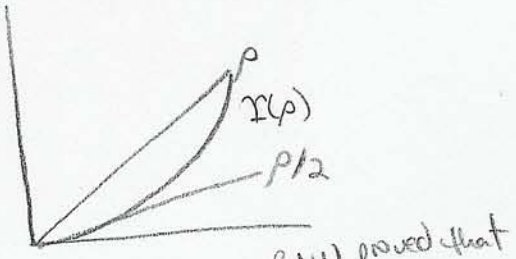
If the maximum margin (for our data & weak classifiers) is ρ, AdaBoost achieves a margin of at least $\rho/2$.



No more contradition with Occam's Razor. But theory still incomplete.

Experimental results often indicated that AdaBoost maximizes the margin — except for Breiman's results. Breiman claimed that his algorithm arc-gv achelved a maximum margin & that AdaBoost did not. (1999)
(But somehow AdaBoost still performed better.)

Rätsch & Warmuth (2005) tightened the bound slightly:



RdW proved that

If max. margin is ρ, AdaBoost achieves margin of at least

$$\Gamma(\rho) = \frac{-\ln(1-\rho^2)}{\ln\left(\frac{1+\rho}{1-\rho}\right)}.$$

It turns out that Rätch & Warmuth were right!

Recently, it has been shown that:

1) AdaBoost does not necessarily maximize the margin
(R, Daubechies, Schapire 04)

2) Rätsch & Warmuth's bound is tight, ie that AdaBoost (in a special case) can achieve a margin of exactly $\Gamma(\rho)$.
(R, Schapire, Daubechies 06, 07)

It also turns out that Breiman was right!

1) Arc-gv maximizes the margin. (Breiman, Rätch & Warmuth) with a fast convergence rate (R, Schapire Daubechies 07)

2) AdaBoost still beats arc-gv (Breiman, Reyzin & Schapire 06) expernmentally

In fact, there have been a number of other algorithms (with fast convergence rates) designed to maximize the margin, none of which has been shown to beat AdaBoost experimentally.

arc-gv (Breiman 99)

AdaBoost-ρ & AdaBoost* (Rätsch & Warmuth 05)

Approx. Coord. Ascent Boosting (R, Schapire, Dasgupta 07)

LP AdaBoost (Grove & Schuurmans 98)

There are many unsolved problems - this is the current state of theoretical research on generalization of AdaBoost!

Experimental work - there are "lots" of variations of AdaBoost (almost all heuristically based, all claim to beat AdaBoost).

One more important fact about AdaBoost:

It solves the machine learning problem of bipartite ranking at the same time as it solves the problem of classification. It was noticed experimentally that AdaBoost is very good at ranking (Cortes & Mohri, Caruana & Niculescu-Mizil 06).

Schapire et al (03) designed a ranking analogue for AdaBoost, called "RankBoost". Turns out AdaBoost is just as good for ranking as RankBoost. In fact, AdaBoost minimizes RankBoost's objective function. (R, Cortes, Mohri, Schapire 06, R S 08)

Next thurs: Formal def of machine learning problem of ranking.