Boosting

Sometimes it is:

- easy to come up with simple, easy to use, rules of thumb classifiers
- but hard to come up with a single highly accurate rule.

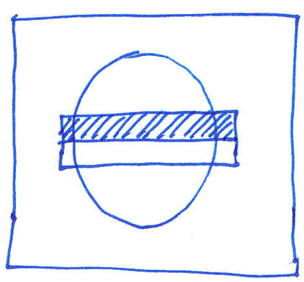
Examples:

(1) Spam classification, based on email text.

Certain words, eg. "Nigeria", "Online Pharmacy", etc. typically are a good indicator of spam.

Rule-of-thumb: Does email contain word "Nigeria"?

(2) Detect if an image has a face in it.



On an average, pixels around the eyes are darker than those below.

Rule of thumb: Is the (average darkness in the shaded region) − (average darkness in the white rectangular region below) > 0 ?

Boosting gives us a way to combine these ~~weak~~ rules ~~into~~ of thumb into good classifiers.

Definitions:

1. Weak Learner: A simple rule of thumb that doesn't necessarily work very well.
2. Strong Learner: A good classifier (with high accuracy)

Boosting Procedure:

1. Design method to find a good rule of thumb.
2. Repeat:
   - Find a good rule of thumb
   - Modify training data to get a second data set
   - Apply method ~~of~~ to new data set to get a good rule of thumb, and so on.

1. How to get a good rule of thumb? Application specific (more later)
2. How to modify training data set?
   - Give highest weight to the hardest examples – those that were misclassified more often by previous rules of thumb.
3. How to combine the rules of thumb into a prediction rule?

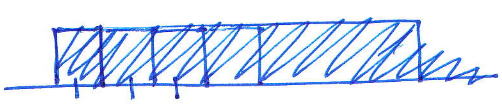
   Take a weighted majority of the rules.

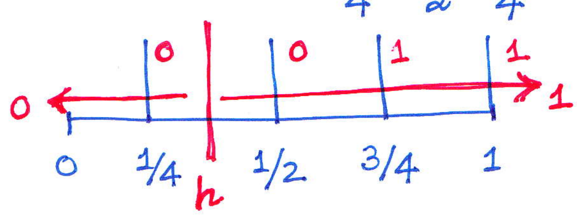
Let D be a distribution over labelled examples, and let h be a classifier.

Error of h wrt D is:

$$err_D(h) = \Pr_{(x,y)\sim D}[h(x) \neq y]$$

Example: D: X: takes values $\frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1$, each w.p. $1/4$.





$Y = 1$ if X has ~~to~~ value $> 1/2$, o/w $Y = 0$.

Then if h is the rule:

$$h(x) = 1 \text{ if } x > 1/4$$
$$= 0 \text{ o/w.}$$

Then, $err_D(h) = \frac{1}{4}$.

→ h is called a <u>weak learner</u> if $err_D(h) < 0.5$

→ Error of random guessing is 0.5 (with 2 labels)

Given training examples $(x_1, y_1), \ldots, (x_n, y_n)$, we can assign weights $w_1, \ldots, w_n$ to these examples. If $\sum_{i=1}^{n} w_i = 1$, $w_i \geq 0$, we can think of these weights as a probability distribution over the examples.

Error of a classifier h wrt W is:

$$err_W(h) = \sum_{i=1}^{n} w_i \, 1(h(x_i) \neq y_i)$$

1 is the indicator function, where $1(P) = 1$ if P is true
$= 0$ otherwise.

<u>Boosting Algorithm:</u>

<u>Input</u>: Training set $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, $y_i = \pm 1$

$D_1(i) = 1/n$ for all $i = 1, \ldots, n$

For $t = 1, 2, 3, \ldots$

h_t = weak-learner wrt D_t. (so, $err_{D_t}(h_t) < 0.5$)

$\varepsilon_t = err_{D_t}(h_t)$

$$\alpha_t = \frac{1}{2} \ln \frac{1-\varepsilon_t}{\varepsilon_t}$$

(so, α_t is high when ε_t is low, and almost 0 when ε_t is close to 0.5

$$D_{t+1}(i) = \frac{D_t(i)\, e^{-\alpha_t y_i h_t(x_i)}}{Z_t}$$

(D_{t+1} goes ↑ if i is misclassified by h_t; so higher D_t means harder example.

where Z_t is a normalization constant to ensure that

$$\sum_i D_{t+1}(i) = 1.$$

Final classifier: $H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$ (weighted majority of $h_t(x)$'s)

Example of Weighted Error:

Suppose training data is: $((0,0),1)$, $((1,0),1)$, $((0,1),-1)$

weights W: $\frac{1}{2}$ $\frac{1}{4}$ $\frac{1}{4}$

classification rule: Predict 1 if $x_1 \leq 1/2$, -1 otherwise.

$$err_W(h) = \frac{1}{2}\times 0 + \frac{1}{4}\times 1 + \frac{1}{4}\times 1 = \frac{1}{2}$$

(The usual (unweighted) error would be $2/3$).

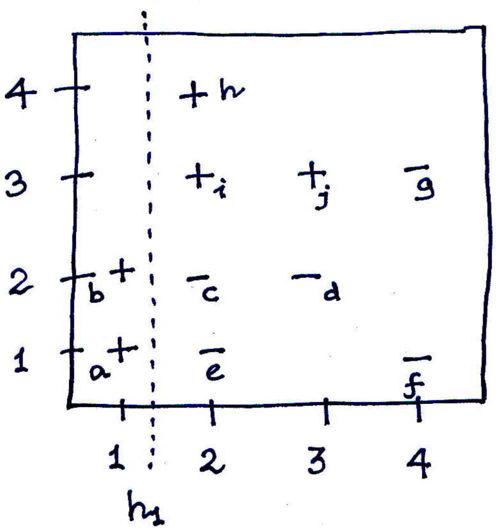
Boosting Algorithm Example:

Training data: $((1,1),+)$ $((2,1),-)$ $((4,1),-)$

$((1,2),+)$ $((2,2),-)$ $((3,2),-)$

$((2,3),+)$ $((3,3),+)$ $((4,3),-)$

$((2,4),+)$



Initially: $D_1(i) = 0.1$ (for all i)

~~Suppose~~

Weak Learners: Set of vertical and horizontal thresholds.

(1) Suppose we pick $h_1(x) = +$ if $x_1 \leq 1.5$

$= -$ otherwise

Name the points: a, b, .., j (for ease of understanding)

Then:

$err_{D_1}(h_1) = \varepsilon_1 = 0.3$ $\quad \alpha_1 = 0.42$

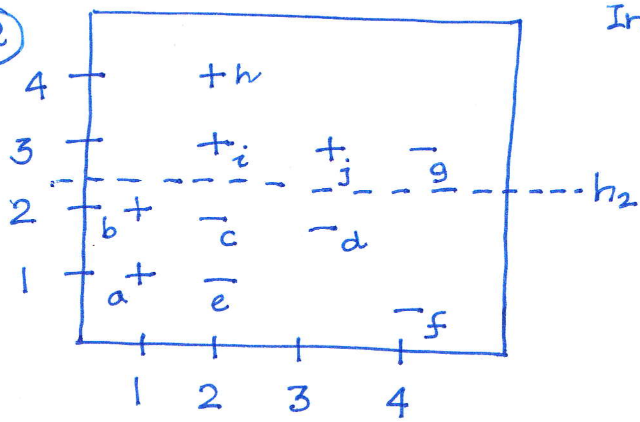
Weights of a, b, c, d, e, f, g: $D_2 = 0.07$

Weights of h, i, j: $D_2 = 0.17$

$$Z_2 = 7 \cdot e^{-0.42} \cdot 0.1 + 3 \cdot 0.1 \cdot e^{0.42}$$
$$= 0.92$$

Note: Calculations rounded to 2 decimal places.

2



In Round 2, suppose we pick

$$h_2(x) = + \text{ if } x_2 > 2.5$$
$$= - \text{ otherwise.}$$

$$err_{D_2}(h_2) = \varepsilon_2 = 0.21$$

$$\alpha_2 = 0.66$$

Weights of a, b: $D_3 := 0.07 \times e^{0.66} / Z_3 = 0.17$

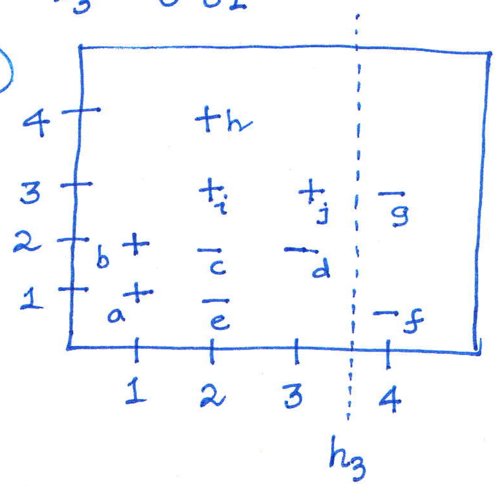
Weights of c, d, e, f: $D_3 := 0.07 \times e^{-0.66} / Z_3 = 0.04$

Weights of h, i, j: $D_3 := 0.17 \times e^{-0.66} / Z_3 = 0.11$

Weight of g: $D_3 := 0.07 \times e^{0.66} / Z_3 = 0.17$

$Z_3 = 0.81$

3



In Round 3, suppose we pick:

$$h_3(x) = + \text{ if } x_1 \leq 3.5$$
$$= - \text{ otherwise.}$$

$$err_{D_3}(h_3) = \varepsilon_3 = 0.12$$

$$\alpha_3 = 0.99$$

Weights of a, b: $D_4 := 0.17 \times e^{-0.99} / Z_4 = 0.1$

" " c, d, e: $D_4 := = 0.04 e^{0.99} / Z_4 = 0.17$

" " h, i, j: $D_4 := 0.11 \times e^{-0.99} / Z_4 = 0.06$

" " f : $D_4 := 0.04 e^{-0.99} / Z_4 = 0.02$ $Z_4 = 0.65$

" " g : $D_4 := 0.17 e^{-0.99} / Z_4 = 0.1$

Final classifier: $\text{sign}(\alpha_1 h_1(x) + \alpha_2 h_2(x) + \alpha_3 h_3(x))$

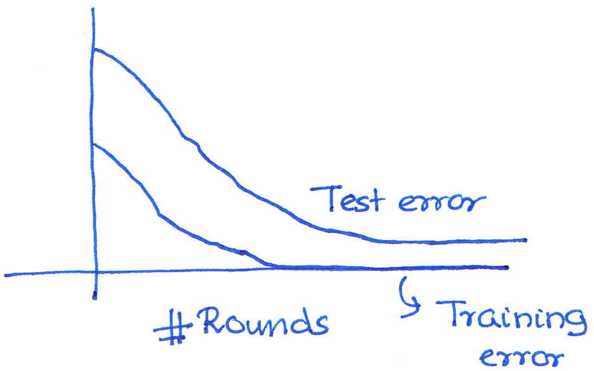
$= \text{sign}(0.42 h_1(x) + 0.66 h_2(x) + 0.99 h_3(x))$

When to stop boosting? Use a validation dataset to find a stopping time. Stop when validation error does not improve.

Boosting and Overfitting:

Overfitting can happen with boosting, but often does not.

Typical boosting run:



Reason is that the margin of classification often increases with boosting.

Intuitively, margin of classification measures how far the + labels are from the − labels.

Small Margin   Large Margin

Note: Notion of margin for boosting is a little different from the exact way we defined margin for perceptron, but the difference is fairly technical.

For boosting:

- think of each $h_t()$ as a feature
- Feature space is:

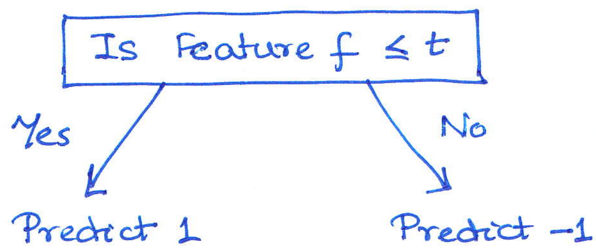
$$[h_1(x), h_2(x), \ldots, h_T(x)]$$

- Margin of example x is: $\left|\sum_{t=1}^{T} \alpha_t h_t(x)\right|$.
- If you have large margin data, then classifiers need less training examples to avoid overfitting. (This is also why kernels work, even if they are very high dimensional feature spaces.)

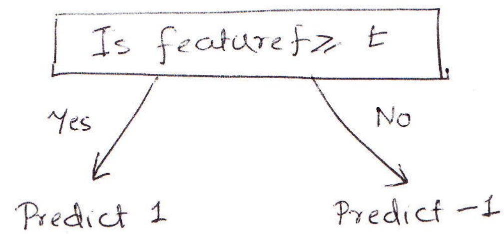
Applications of Boosting:

1. Boosted Decision trees:

Weak learners are single node decision trees of the form:



or



2. Face detection: Viola and Jones: see slides.