Boosting, a Form of Ensemble Learning

"rules of thumb"

motivation: sometimes it is easy to come up with simple rules that perform somewhat okay (accuracy 60%, or 55%, or even just barely above 50%)

but hard to come up with a single all-encompassing highly accurate rule.

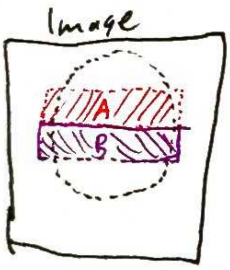
Ex: Spam classification. A full predictor sounds hard, but we can design a lot of simple rules:

"Email has "online pharmacy" in it?" → spam
else → not spam

might not be super accurate, but likely a fair bit better than random guessing!

Ex: Face detection. What exactly is a face or not a face?

But again, we might make simple rules that are better than nothing:



Broadly speaking, pixels around eyes are in shadow, and generally darker than the pixels below.

Simple rule: If (avg. darkness in box A) − (avg. darkness in box B) > 0
then predict "face"
else predict "not face"

Ensemble Learning: An ensemble is a collection of classifiers, where the overall prediction is formed using predictions from the multiple classifiers in the ensemble.

Boosting is one particular approach for choosing good rules to add to an ensemble, and a good way to mix its predictions with the others.

Definition

1] Weak learner: a predictor that at least works better than random guessing. (ex. our simple rules from before)

2] Strong learner: a good predictor, with desirable levels of accuracy for a task.

Can a strong learner also be a weak learner?

Boosting (High-Level)

1] Assume we have a way of finding decent weak learners

2] Repeat:

- Find a decent weak learner for our training data
- Add that weak learner to our ensemble
- Modify our training data, get a new training set for the next round.
- repeat until we are satisfied

3] Output our ensemble as a strong learner.

How do we make a weak learner? That's application specific

How do we add a weak learner to our ensemble?

We pick a weight, and our overall prediction is just a weighted majority.

How do we modify the data?

We change the importance, or weight, of each example.

Points we keep getting wrong become more important (the hardest examples)

Points we have been consistently correct on become less important.

we want to select a new rule that covers or fixes the mistakes of our past rules.

In math and theory: when we change rounds, we want to change the effective distribution ~~of~~ of our training data, such that the difficult points where our ensemble has been making mistakes receive more representation.

another equivalent view

$$\text{training error} = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}(f(x_i) \neq y_i) = \Pr_{i \sim \text{Uniform}([n])}\left(f(x_i) \neq y_i\right)$$

Uniformly select an index from our training set.

It is this probabilistic view where the theory of boosting arises.

In practice: We are going to define a weighted training error

Let $S = (x_1, y_1), \dots, (x_n, y_n)$ be our training set

Let $w = \{w_1, \dots, w_n\}$ be our weights, where

- $w_i \geq 0$ for all i in $1, \dots, n$
- $\sum_{i=1}^{n} w_i = 1$

these constraints ensure this last view makes sense

then $$err_w(f) = \sum_{i=1}^{n} w_i \mathbb{1}(f(x_i) \neq y_i) = \Pr(f(x_i) \neq y_i)$$

when $\Pr(i=j) = w_j$

f is called a weak learner w.r.t. w if $err_w(f) < 0.5$

Example: Data is $((0,0),1), ((1,0),1), ((0,1),-1)$ (assuming just 2 labels)

weights w: $1/2, 1/4, 1/4$

$$f(x) = \begin{cases} 1 & \text{if } x^{(1)} \leq 1/2 \\ -1 & \text{otherwise} \end{cases} \quad err_w(f) = \frac{1}{2}\cdot 0 + \frac{1}{4}\cdot 1 + \frac{1}{4}\cdot 1 = 1/2$$

correct on first point / wrong on second / wrong on third

Boosting Algorithm (Detailed)

like perceptron, the math only works if we assume positive vs negative labels.

Input: training Data $(x_1, y_1) \cdots (x_n, y_n)$, Assume $y_i = +1$ or -1

1] Initialize our distribution over training data: $D_1(i) = 1/n$ for all $i = 1, \ldots, n$

(init. to a uniform distribution)

D_1: subscript for current round; i: index of training point associated with this weight.

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

- h_t = (select a weak learner for D_t)
- $\varepsilon_t = err_{D_t}(h_t)$
- $\alpha_t = \frac{1}{2} \ln \frac{1-\varepsilon_t}{\varepsilon_t}$

For all i:

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

where Z_t is a normalization constant, (we make $\sum_i D_{t+1}(i) = 1$ after all the scaling)

- Like perceptron, we can go as long as we want or can.
- h_t, can be any weak learner for our current weights, $err_{D_t}(h_t) < 0.5$
- we will use ε_t, the weighted error to compute α_t
- α_t is the importance h_t will receive in the final ensemble
  (ε_t low $\longleftrightarrow$ h_t accurate $\longleftrightarrow$ α_t big
  ε_t high $\longleftrightarrow$ h_t inaccurate $\longleftrightarrow$ α_t small)
- reweight each data point.

Case 1: $y_i = h_t(x_i)$, h_t correct

$-\alpha_t y_i h_t(x_i) < 0$

$\Rightarrow$ we scale down the weight

Case 2: $y_i \neq h_t(x_i)$, h_t wrong

$-\alpha_t y_i h_t(x_i) > 0$

$\Rightarrow$ we scale up the weight

THUS: D_{t+1} gives more importance to the harder examples

3] Output our final classifier (strong learner), a weighted majority of weak learners:

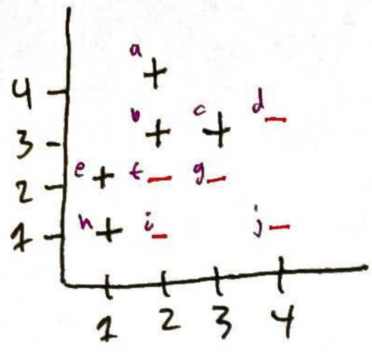
$$f(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

$h_t(x)$: weak learner chosen in round t

α_t: the importance of that weak learner in our ensemble

Boosting Example

Data $((1,1),+)$ $((2,1),-)$ $((4,1)\ -)$ $((1,2),+)$
$((2,2)\ -)$ $((3,2),-)$ $((2,3),+)$ $((3,3),+)$
$((4,3),-)$ $((2,4),+)$



lettered for convenience,
we have 10 points: a,b,c,d,e,f,g,h,i,j

$\boxed{\text{Init:}}$ $D_1(i) = 0.1$ for all i. (uniform weights over 10 points)

Weak learners: vertical or horizontal thresholds

$\boxed{\text{Exercise: How many distinct weak learners do we have here?}}$

Suppose we pick h_1 to be: $\begin{cases} + & \text{if } x_{(1)} \le 1.5 \\ - & \text{otherwise} \end{cases}$

then h_1 gets e, h, d, f, g, i, j correct
and a, b, c incorrect

$\varepsilon_1 = err_{D_1}(h_1) = 3/10$, and $\alpha_1 \approx 0.42$

$\boxed{\text{make sure to use the natural log.}}$

Time to reweight:

e, h, d, f, g, i, j weights go <u>down</u>, scaled by $e^{-\alpha_1}$

a, b, c weights go up, scaled by $e^{+\alpha_1}$

<u>and</u> we must make the new weights sum to 1.

Z_1 = sum of our unnormalized weights $= 7\cdot(0.1\cdot e^{-0.42}) + 3\cdot(0.1 e^{+0.42})$
≈ 0.92

$\Rightarrow D_2$ gives weight 0.07 to points e, h, d, f, g, i, j
and weight 0.17 to points a, b, c

$\boxed{0.07 < 0.1 < 0.17}$
starting weights

That's Round 1. Moving onto round 2:

Suppose we pick h_2 to be: $\begin{cases} + & \text{if } x_{(2)} > 2.5 \\ - & \text{otherwise} \end{cases}$

then h_2 gets a, b, c, f, g, i, j correct
and d, e, h incorrect.

$\varepsilon_2 = err_{D_2}(h_2) = 0.07 + 0.07 + 0.07 = 0.21$

$\alpha_2 = 0.66$

weights of e, h: were 0.07, go up by $e^{+\alpha_2}$

weights of f, g, i, j: were 0.07, go down by $e^{-\alpha_2}$

weights of a, b, c: were 0.17, go down by $e^{-\alpha_2}$

weight of d: was 0.07, goes up by $e^{+\alpha_2}$

$$Z_3 = 3\left(0.07e^{0.21}\right) + 4\left(0.07e^{-0.21}\right) + 3\left(0.17e^{-0.21}\right)$$

$$\approx 0.81$$

$\Rightarrow D_3$ gives weight 0.17 to d, e, h ← points correct in first round, wrong in second

weight 0.11 to a, b, c ← points wrong in first round, correct in second

weight 0.04 to f, g, i, j ← points that were correct in both rounds

Round 3 (abbreviated): Suppose $h_3 = \begin{cases} + & \text{if } x_1 \leq 3.5 \\ - & \text{otherwise} \end{cases}$

correct	incorrect
abcdehj	fgi

$\varepsilon_3 = err_{D_3}(h_3) = 0.12 \qquad \alpha_3 = 0.99$

$D_4 =$

a	b	c	d	e	f	g	h	i	j
0.06	0.06	0.06	0.1	0.1	0.17	0.17	0.1	0.17	0.02

j has the smallest weight in D_4,
also: j was correctly predicted in all 3 rounds.

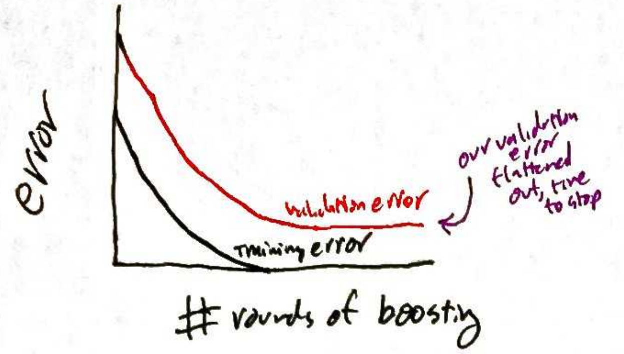
If we stop after round 3, our final classifier is:

$$f(x) = \text{sign}\left(\alpha_1 h_1(x) + \alpha_2 h_2(x) + \alpha_3 h_3(x)\right) = \left(0.42\, h_1(x) + 0.66\, h_2(x) + 0.99\, h_3(x)\right)$$

When to stop boosting?

Whenever we are satisfied. One common approach is to use a validation data set, and measure the validation error over time.

We stop when it looks like our validation error does not improve.

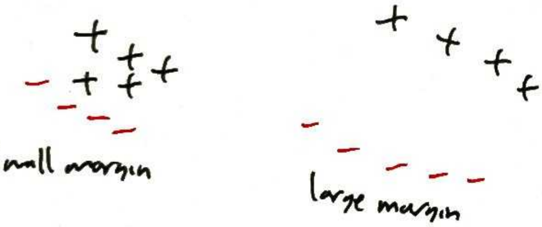


Boosting and Overfitting

Overfitting can happen with boosting
(true error goes up with more rounds)

but often it **does not**. In practice we tend to see the validation error just stay flat after a while.

Why? The theory is that adding more weak learners to our ensemble usually increases the margin of classification. Here we refer to a measure of how far the + labels are from the − labels.



Note: this notion of margin for boosting is a little different from the precise way we defined margin for perceptrons, but the difference is fairly technical.

Idea:
- think of each weak learner $h_t(x)$ as a feature for x [either +1 or −1]
- our feature space is $[h_1(x), h_2(x), \ldots, h_T(x)]$
- the margin of example x is $\left|\sum_{i=1}^{T} \alpha_t h_t(x)\right|$.
- With high margins, we need fewer points to avoid problems with overfitting.
  (This idea also is why Kernel methods with huge feature spaces can still work with reasonable amount of data)

Popular Applications of Boosting:

1. Boosted Decision Trees.
   "Decision stumps", or trees with only 1 node, are a common class of weak learners
2. Face detection. Viola-Jones made real time face detection possible with a boosting approach.