AdaBoost and RankBoost

Josiah Yoder

School of Electrical and Computer Engineering

RVL Seminar, 4 Feb 2011

Overview

- ROC & Precision-Recall Curves
- AdaBoost
- RankBoost
- Not really any of my research

ROC & Precision-Recall

ROC – Receiver Operating Characteristics

- Good with skewed Distribution
- Good with unequal costs
- Good for visualizing maximum performance with any given threshold

Precision-Recall Curve

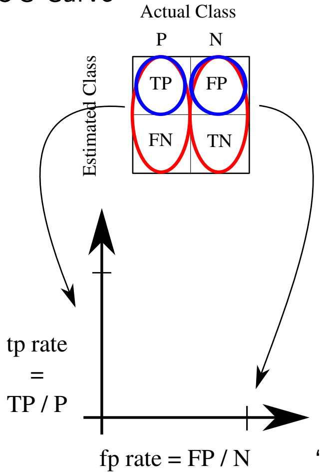
- Good for evaluating ranked search results
- Consistent for comparing performance

Adapted from Fawcett (2003}

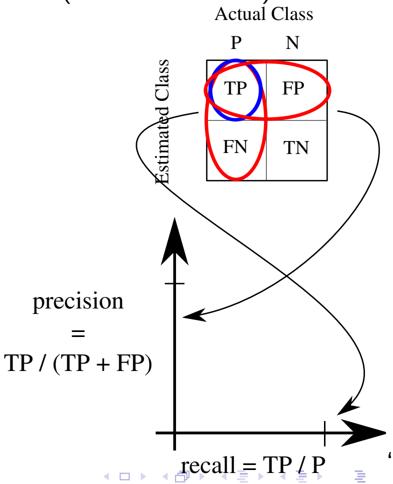
		Actual Class	
		P	N
Estimated Class	P	True Positive	False Positive
	N	False Negitive	True Negitive

The Axes

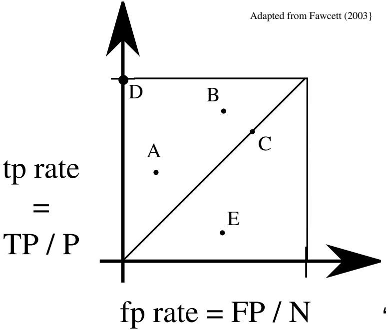
ROC Curve



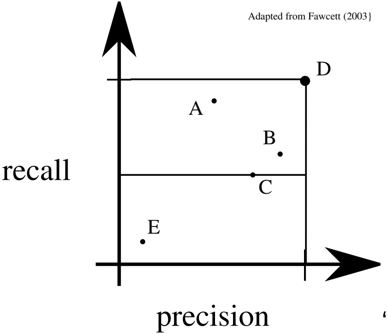
P-R (Precision-Recall) Curve



ROC Points



P-R (Precision-Recall) Points

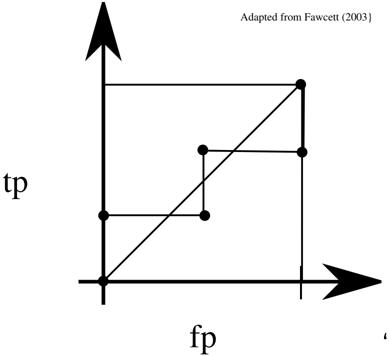


(50% P, 50% N)

Numeric Classifiers

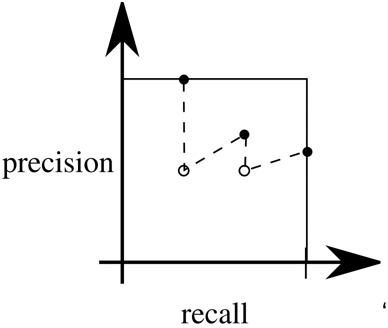
- Output: Numeric value (score, probability, rank ...)
- Thresholding output gives a discrete score — single point on ROC or PR curve
- "Sweeping" the threshold from ∞ to $-\infty$ traces out a curve
- There is a more efficient way to do it
- Note that ROC and PR curves are parametric

ROC Curve



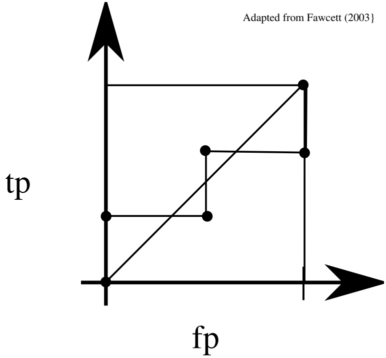
GT	1	0	1	0	1
feat	0.9	0.7	0.6	0.5	0.2

P-R (Precision-Recall) Curve

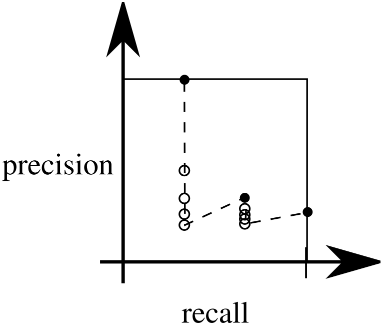


(50% P, 50% N)

ROC Curve



P-R (Precision-Recall) Curve



(20% P, 80% N)

GT	1	0	0	0	0	1	0	0	0	0	1
feat	0.9	0.7	0.7	0.7	0.7	0.6	0.5	0.5	0.5	0.5	0.2

Which is better?

- ROC doesn't change if ratio P/N changes
  - Makes it easy to visualize maximum performance when ratio of true to false changes
- P-R illustrates search results
  - Clearly illustrates effect of false positives on performance

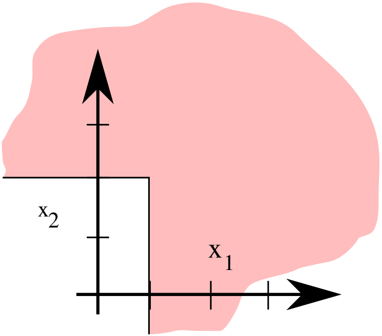
AdaBoost and RankBoost

- AdaBoost
- RankBoost
- Classification
- Ranking
- ROC
- P-R

- What is Boosting?
  - Linear combination of weak classifiers with proven bounds on performance
  - A weak classifier is one that gets more than random guessing right.
- AdaBoost
  $h(x)$ — weak binary classifier, $h : \mathscr{X} \to \{-1, 1\}$
  $H(x)$ — strong classifier, $H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

Nonlinear separation

- Combination of weak classifiers is linear, but weak classifiers are non-linear
- Example
  - $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$
  - $H(x) = sign(0.5[[x_1 > 1]] + 0.5[[x_2 > 2]])$

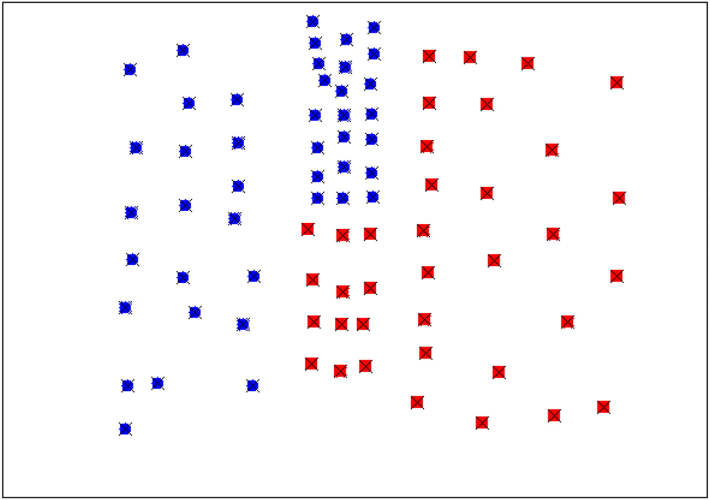


How to pick weak classifiers and their weights

- Greedy: Pick the best weak classifier repeatedly
- Focus on the points misclassified by the previous classifier
  - (Not on overall performance)
- Following frames from http://cseweb.ucsd.edu/~yfreund/adaboost/index.html

prediction sum ☑ Training set

prediction rule ☐ Test set



edit

clear

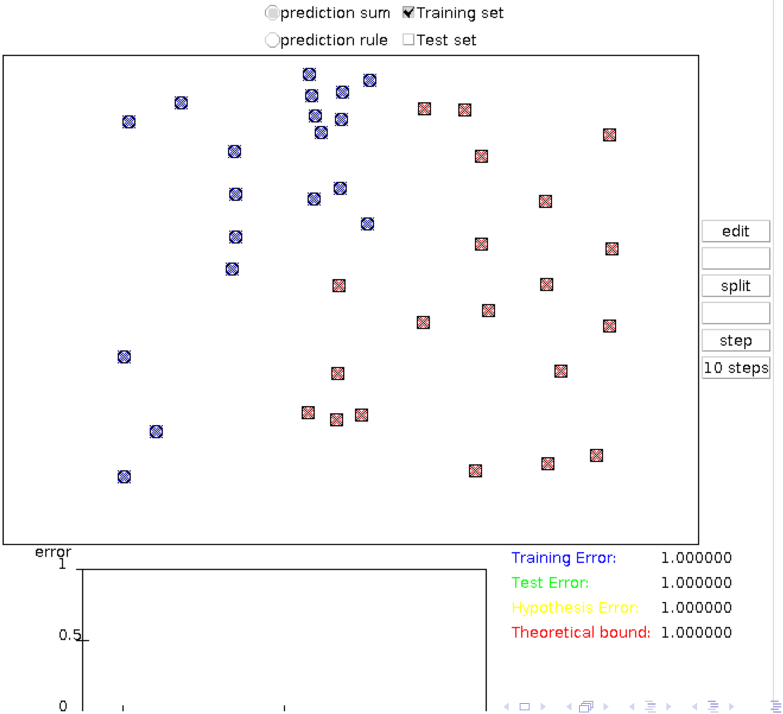
split

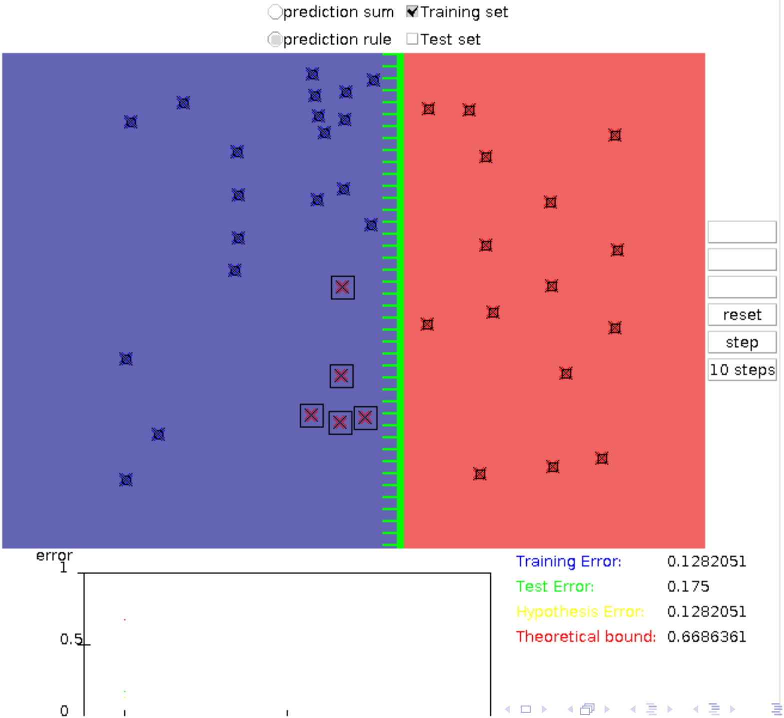
Training Error: 1.000000

Test Error: 1.000000

Hypothesis Error: 1.000000

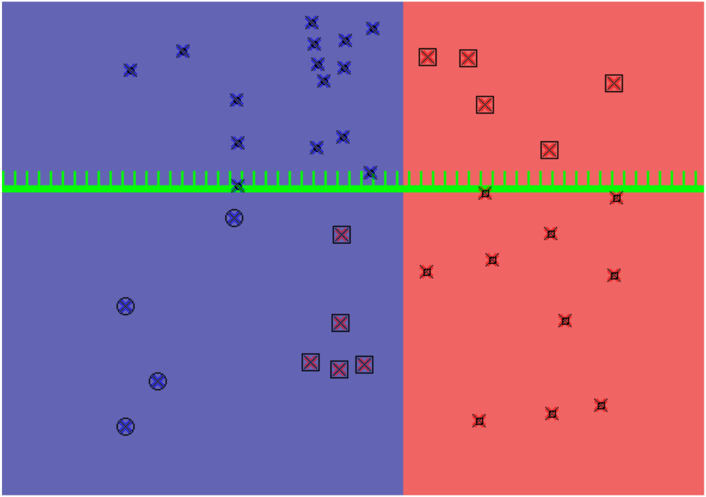
Theoretical bound: 1.000000


prediction sum
Training set
prediction rule
Test set
edit
split
step
10 steps
error
1
0.5
0
Training Error: 1.000000
Test Error: 1.000000
Hypothesis Error: 1.000000
Theoretical bound: 1.000000


prediction sum
Training set
prediction rule
Test set
reset
step
10 steps
error
1
0.5
0
Training Error: 0.1282051
Test Error: 0.175
Hypothesis Error: 0.1282051
Theoretical bound: 0.6686361

prediction sum ☑ Training set

prediction rule ☐ Test set

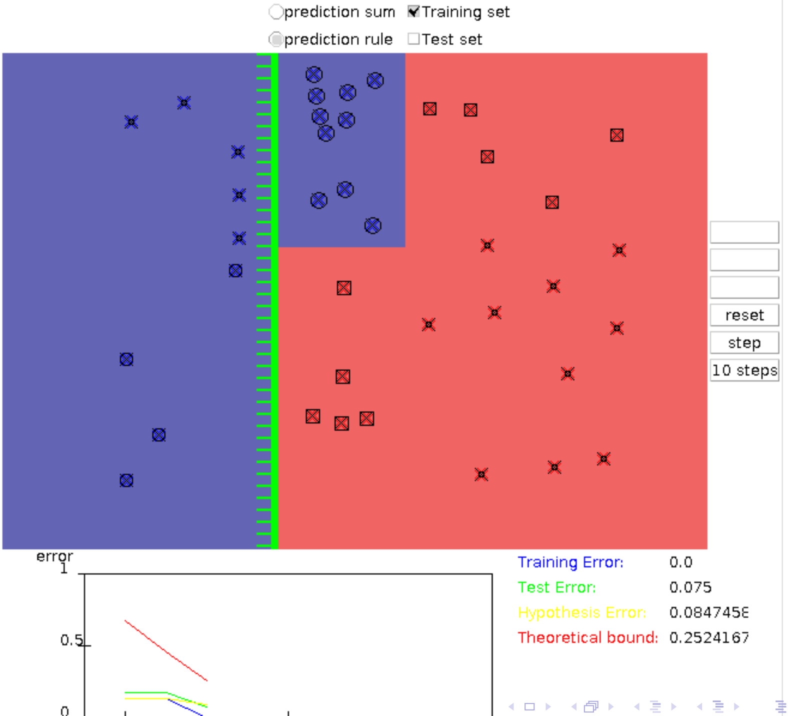


reset

step

10 steps

Training Error:	0.1282051
Test Error:	0.175
Hypothesis Error:	0.1323528
Theoretical bound:	0.4531667


prediction sum
Training set
prediction rule
Test set
reset
step
10 steps
error
1
0.5
0
Training Error: 0.0
Test Error: 0.075
Hypothesis Error: 0.0847458
Theoretical bound: 0.2524167

How to reweight the training data

- How to reweight the training data
  - $D_1 = 1/m$ where m is the number of training points.
  - $D_{t+1}(i) = \frac{D_t(i) \cdot \spadesuit}{Z_t}$
- What to use for $\spadesuit$? (dropping t subscript for a while...)
  - Idea #1: $\spadesuit = [[y_i \neq h(x_i)]]$
    - Problem: correctly classified data is totally forgotten
  - Idea #2: $\spadesuit = \begin{cases} a & y_i \neq h(x_i) \\ 1/a & y_i = h(x_i) \end{cases}$
    - Tada! This is what AdaBoost does!

ADABOOST'S ♠

- Idea #2: $\spadesuit = \begin{cases} a & y_i \neq h(x_i) \\ 1/a & y_i = h(x_i) \end{cases}$
- AdaBoost: $\spadesuit = \begin{cases} e^{\alpha} & y_i \neq h(x_i) \\ e^{-\alpha} & y_i = h(x_i) \end{cases}$
  - or equivalently, $\spadesuit = \exp[-\alpha y_i h(x_i)]$
- What is α? How do you choose it? We will come back to this...
  - We shall see that the classifier has nice properties no matter how we choose α

The Basic AdaBoost Algorithm

(t subscripts are back)

- $D_1 = 1/m$
- for $i = 1, \ldots, T$
  - Choose $h_t(x_i)$ based on data weighted by D_t
  - Reweight

$$D_{t+1}(i) = \frac{D_t(i) \cdot exp[\alpha_t y_i h_t(x_i)]}{Z_t}$$

$$Z_t = \sum_i D_t(i) \cdot exp[-\alpha_t y_i h_t(x_i)]$$

- $H(x) = sign\left(\sum \alpha_t h_t(x_i)\right)$
- $f(x) = \sum \alpha_t h_t(x_i)$

Why can we just use α as the weight?

- It allows a nice bound on the error.

$$\frac{1}{m}\sum_i [[H(x_i) \neq y_i]] \leq \frac{1}{m}\sum_i \exp\left[-y_i f(x_i)\right]$$

$$[[H(x_i) \neq y_i]] \leq exp\left[-y_i f(x_i)\right]$$

$$[[f(x_i)y_i \leq 0]] \leq exp\left[-y_i f(x_i)\right]$$

$$\frac{1}{m}\sum_i [[H(x_i) \neq y_i]] \leq \frac{1}{m}\sum_i \exp\left[-y_i f(x_i)\right] = \prod_t Z_t$$

- That looks pretty. But what does it mean?
  - Z_t is the normalizing constant for the weights on each point
  - Roughly, $Z_t \approx \sum_i D_t(i)[[y_i \neq h_t(x_i)]]$, the cost of misclassifying the weighted points in the t^{th} round.

THAT'S A PRETTY NICE BOUND.

- How do we know it's true?

$$D_4(i) = \frac{\frac{\frac{D_1 exp[\alpha_1 y_i h_1(x_i)]}{Z_1} exp[\alpha_2 y_i h_2(x_i)]}{Z_2} exp[\alpha_3 y_i h_3(x_i)]}{Z_3}$$

$$D_{T+1}(i) = \frac{D_1 \prod_t \exp[-\alpha_t y_i h_t(x_i)]}{\prod_t Z_t} = \frac{\frac{1}{m} \exp[\sum_t -\alpha_t y_i h_t(x_i)]}{\prod_t Z_t} = \frac{\frac{1}{m} exp[y_i f(x_i)]}{\prod_t Z_t}$$

$$1 = \sum_i D_{T+1}(i) = \frac{1}{m} \sum_i \exp[y_i f(x_i)] / \prod_t Z_t$$

$$\prod_t Z_t = \frac{1}{m} \sum_i \exp[y_i f(x_i)] \qquad \text{so} \qquad \text{err}_{train} \leq \prod_t Z_t$$

We can almost implement this. Now what about α?

- Choose α_t to minimize Z_t

$$\frac{\partial}{\partial \alpha} Z_t = \dots$$

- $\alpha_t = 2 \ln \left(\frac{1-\varepsilon_t}{\varepsilon_t} \right)$ where ε_t is the weighted error of the classifier at the t_{th} stage
- And to compute the weak learners, create an ROC curve using each dimension of the data alone. Take the best point on the ROC curve and use that as your classifier.
- This completes the algorithm.
- Thinking back to ROC curves:
  - The goal of AdaBoost is to minimize the classifier error. Same thing as trying to make the best performance of the ROC curve as close to the top left corner as possible.

RankBoost

- Goal is to find ordering, not "quality" of each point.
- Does not attempt to directly maximize MAP, but is successful at doing this anyhow
- Error is defined in terms of the number of data pairs which are out of order
- $D(x_0, x_1) = c$ if x_0 should be ranked below x_1, $D(x_0, x_1) = 0$ otherwise.
- $\sum_{x_0, x_1} D(x_0, x_1) = 1$
- More complicated D can be use to emphasize really important pairs.

EXAMPLE

e.g. 1. True rank should be $rank(x_1) < rank(x_2) < rank(x_3)$.

D	x_1	x_2	x_3
x_1	0	1/3	1/3
x_2	0	0	1/3
x_3	0	0	0

e.g. 2. True rank should be $rank(x_1) < rank(x_2) = rank(x_3)$.

D	x_1	x_2	x_3
x_1	0	1/3	1/3
x_2	0	0	0
x_3	0	0	0

Basic RankBoost Algorithm

Given: Rank matrix D with true ranks of the training data

- $D_1 = D$
- For $t = 1, \ldots, T$
  - Train $h_t : X \to \{-1, 1\}$
  - Choose α_t (more suspense...)
  - Update

$$D_{t+1}(x_0, x_1) = \frac{D_t(x_0, x_1) \exp[\alpha_t(h_t(x_0) - h_t(x_1))]}{Z_t}$$

- Final ranking: $H(x) = \sum_t \alpha_t h_t$

Bounds and α

- As before, $err_{train} \leq \prod_t Z_t$
- Selection of α is different. Now $\alpha = \frac{1}{2}\ln\left(\frac{1+r}{1-r}\right)$ where $r = W_i - W_+ = \sum D(x_0, x_1)(h(x_1) - h(x_0))$ and W_+ is the weight of the pairs for which $h(x_0) > h(x_1)$ and W_- is the weights of the pairs for which $h(x_0) < h(x_1)$.
- Selection of weak classifier also needs to be redone. Turns out we need to maximize r, which can be written as

$$r = \sum_x h(x)s(x)v(x) \sum_{x' \in s(x) \neq s(x')} v(x')$$

where $v(x)$ [SHOULD BE] given above and

$$s(x) = \begin{cases} +1 & x \in X_0 \\ -1 & x \in X_1 \end{cases}$$

Sorry!

I ran out of time!

Acknowledgements

- I wish to thank my advisor, Prof. Avi Kak for the helpful observation that ROC curves and PR curves are parametric. His presentation on retrieval in the summer of 2011 also presents an entertaining comparison of PR-curves for random and ideal retrieval.
- I also wish to thank my lab-members for pointing out errors in this presentation that have hopefully all been corrected in this version.
  - Note that we covered RankBoost in 5 or 10 minutes so errors probably remain on slides 11 through 30.