Approaches to Multiple-Instance Learning

Marina and Robert Langlois

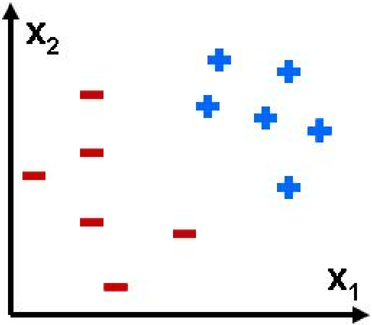
August 8, 2011

Classic Classification problem

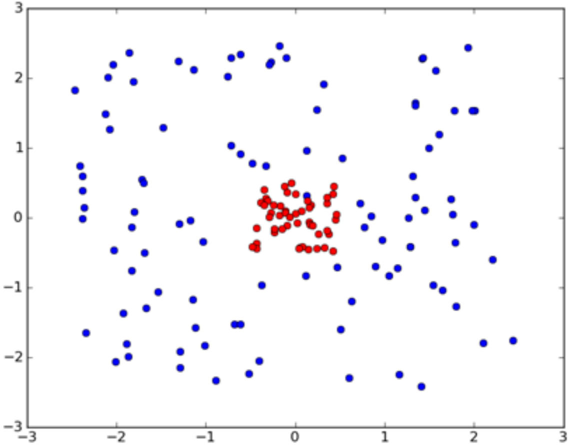
- Let $S =< (x_1, y_1), \ldots, (x_m, y_m) >$ - training examples.
- $x_i \in X$ - instance space and $y_i \in Y$ -finite label space Y.
- Binary classification problems in which $Y = \{-1, 1\}$

- **To find**: Prediction rule or to learn to predict a categorical outcome from input

Pictures of possible classification problems



Pictures of possible classification problems



"How may I help you?" (From R. Schapire talk)

- **goal**: automatically categorize type of call requested by phone customer
  (*Collect*, *CallingCard*, *PersonToPerson*, etc.)
- yes I'd like to place a collect call long distance please (*Collect*)
- yes I'd like to place a call on my master card please (*CallingCard*)
- I just called a number in sioux city and I musta rang the wrong number because I got the wrong party and I would like to have that taken off of my bill (*BillingCredit*)

Example Cont.

Observation:

- Easy: to find "rules of thumb" that are "often" correct
  • e.g.: IF 'card' occurs in utterance THEN predict 'CallingCard'
- Hard: to find **single** highly accurate prediction rule

The Boosting Approach

- Devise a comp. program for deriving rough rules of thumbs.
- select small subset of examples
- derive rough rule of thumb
- examine 2nd set of examples
- derive 2nd rule of thumb
- repeat T times

Questions:

- how to choose subsets of examples to examine on each round?
- how to combine all the rules of thumb into single prediction rule?

• boosting = general method of converting rough rules of thumb into highly accurate prediction rule

Details

- How to choose examples on each round?
  – concentrate on "hardest" examples (misclassified by the prev. rules of thumb)

Details

- How to choose examples on each round?
  – concentrate on "hardest" examples (misclassified by the prev. rules of thumb)
- How to combine rules of thumb into a single prediction rule?
  – Take (weighted) majority vote of rules of thumb.

AdaBoost Approach. When and Who?

- 1995 AdaBoost (Freund and Schapire)
- 1997 Generalized version of AdaBoost (Schapire and Singer)

AdaBoost Approach

- AdaBoost is an algorithm for constructing a "strong" classifier as linear combination of "weak" classifiers $h_t(x)$:

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

- Final hypothesis $H(x) = sign(f(x))$.
- Confidence is $|H(x)|$.

AdaBoost Algrithm

- given training set $(x_1, y_1), \ldots, (x_n, y_n)$, where $y_i \in \{-1, +1\}$ correct label of instance $x_i \in X$.

AdaBoost Algrithm

- given training set $(x_1, y_1), \dots, (x_n, y_n)$, where $y_i \in \{-1, +1\}$ correct label of instance $x_i \in X$.
- initialize weights $w_i = 1/n$

AdaBoost Algrithm

- given training set $(x_1, y_1), \ldots, (x_n, y_n)$, where $y_i \in \{-1, +1\}$ correct label of instance $x_i \in X$.
- initialize weights $w_i = 1/n$
- for $t = 1, \ldots, T$

AdaBoost Algrithm

- given training set $(x_1, y_1), \ldots, (x_n, y_n)$, where $y_i \in \{-1, +1\}$ correct label of instance $x_i \in X$.
- initialize weights $w_i = 1/n$
- for $t = 1, \ldots, T$
- (Call *WeakLearn*) Find weak hypothesis ("rule of thumb")

$$h_t : X \to \{-1, +1\}.$$

with minimum error w.r.t. distribution w_t;

AdaBoost Algrithm

- given training set $(x_1, y_1), \ldots, (x_n, y_n)$, where $y_i \in \{-1, +1\}$ correct label of instance $x_i \in X$.
- initialize weights $w_i = 1/n$
- for $t = 1, \ldots, T$
- (Call *WeakLearn*) Find weak hypothesis ("rule of thumb")

$$h_t : X \to \{-1, +1\}.$$

  with minimum error w.r.t. distribution w_t;
- Choose $\alpha_t \in R$,

AdaBoost Algrithm, cont

- Update:

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

where Z_t is the normalization factor, s.t. w_{t+1} is distribution:

$$Z_t = \sum_{i=1}^{n} w_{t+1}(i).$$

AdaBoost Algrithm, cont

- Update:

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

where Z_t is the normalization factor, s.t. w_{t+1} is distribution:

$$Z_t = \sum_{i=1}^{n} w_{t+1}(i).$$

- Output $H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

Re-weighting

Effect on the training set

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

Re-weighting

Effect on the training set

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

$$e^{-\alpha_t y_i h_t(x_i))} = \begin{cases} < 1, & \text{if } y_i = h_t(x_i) \\ > 1, & \text{if } y_i \neq h_t(x_i) \end{cases}$$

Re-weighting

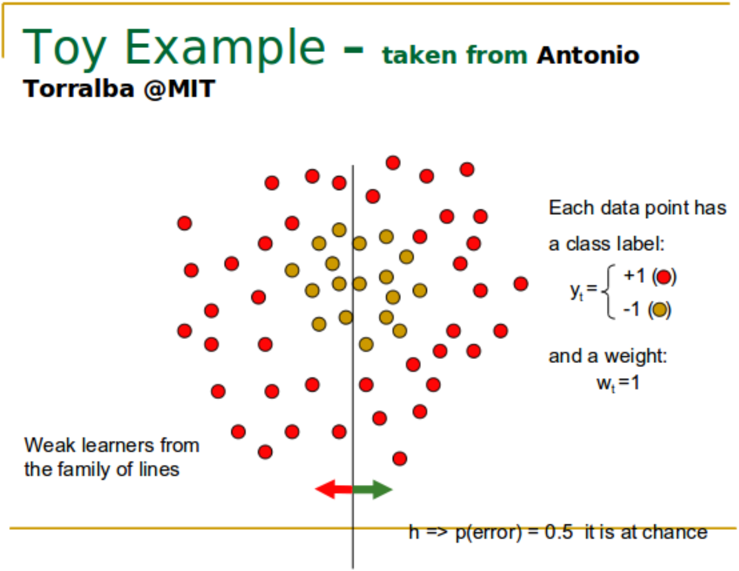
Effect on the training set

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

$$e^{-\alpha_t y_i h_t(x_i))} = \begin{cases} < 1, & \text{if } y_i = h_t(x_i) \\ > 1, & \text{if } y_i \neq h_t(x_i) \end{cases}$$

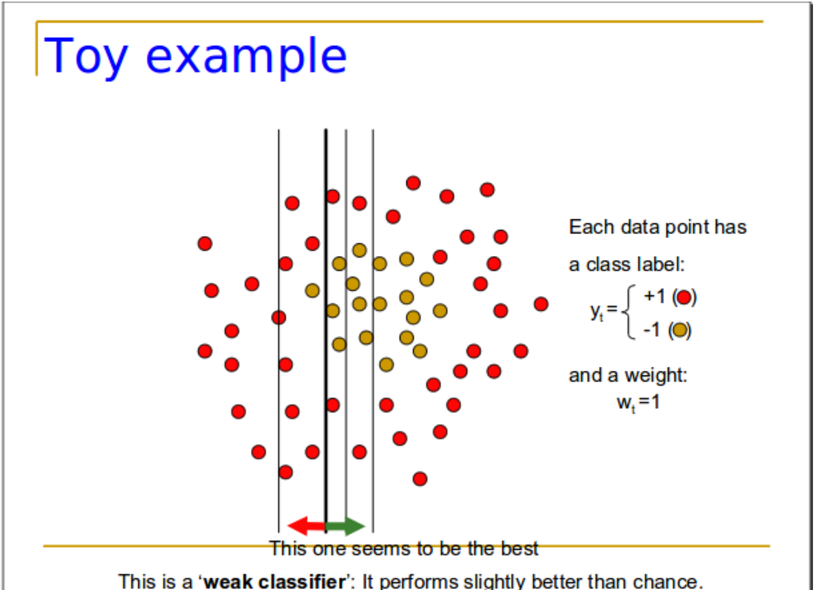
Thus Increase (decrease) weight of wrongly (correctly) classified examples.

Toy example



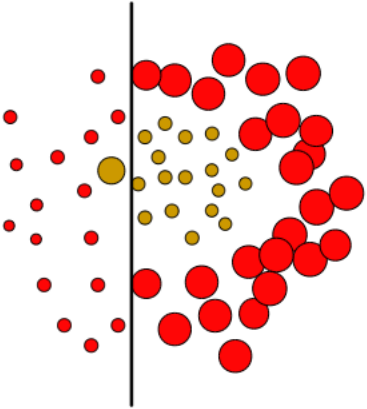
Toy Example - taken from Antonio Torralba @MIT
Each data point has
a class label:
y_t = +1 (●) / -1 (●)
and a weight:
w_t =1
Weak learners from
the family of lines
h => p(error) = 0.5 it is at chance

Toy example


Toy example
Each data point has
a class label:
$y_t = \begin{cases} +1 \\ -1 \end{cases}$
+1
-1
and a weight:
$w_t = 1$
This one seems to be the best
This is a 'weak classifier': It performs slightly better than chance.

Toy example

Toy example



Each data point has

a class label:

$$y_t = \begin{cases} +1 \; (\;) \\ -1 \; (\;) \end{cases}$$

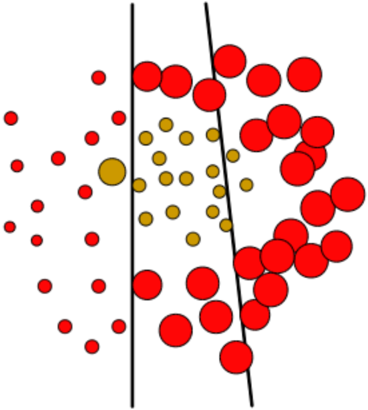
We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t H_t\}$$

We set a new problem for which the previous weak classifier performs at chance again

Toy example

Toy example



Each data point has a class label:

$$y_t = \begin{cases} +1 \ (\text{red}) \\ -1 \ (\text{yellow}) \end{cases}$$

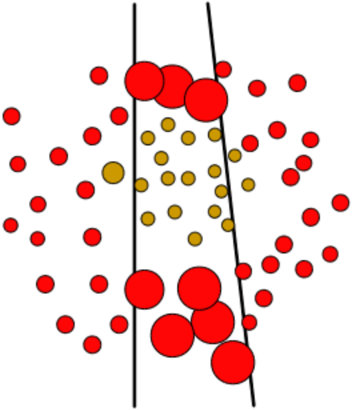
We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t H_t\}$$

We set a new problem for which the previous weak classifier performs at chance again

Toy example

Toy example



Each data point has a class label:

$$y_t = \begin{cases} +1 \ (\bullet) \\ -1 \ (\bullet) \end{cases}$$

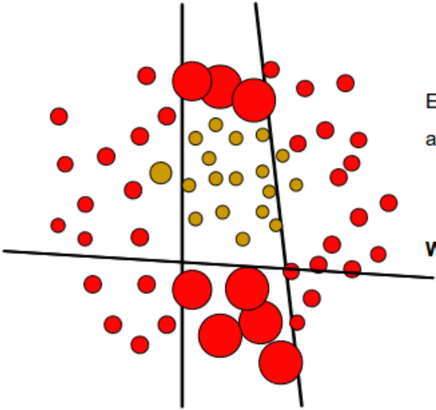
We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t H_t\}$$

We set a new problem for which the previous weak classifier performs at chance again

Toy example

Toy example



Each data point has

a class label:

$$y_t = \begin{cases} +1 \ (\bullet) \\ -1 \ (\bullet) \end{cases}$$

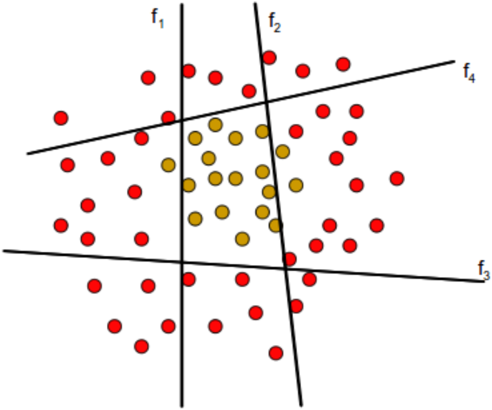
We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t H_t\}$$

We set a new problem for which the previous weak classifier performs at chance again

Toy example

Toy example



The strong (non- linear) classifier is built as the combination of all the weak (linear) classifiers.

Multiple-Instance Problem

New Problem To Solve

Drug activity problem by Dietterich et al

- Example is a molecule and the points that make up the example correspond to different physical configurations of that molecule;
- And the label indicates whether or not the molecule has a desired binding behavior, which occurs if at least one of the configurations has the behavior.

Multiple-Instance Problem. Motivation

- It is not always possible to provide labeled instances for training
- Reasons
  - Requires substantial human effort
  - Requires expensive tests
  - Disagreement among experts
  - Labeling is not possible at instance level
- **Objective**: present a learning algorithm that can learn from limited information.

Multiple-Instance Problem. Formulation.

- In MIL, instead of giving the learner labels for the individual examples, the trainer only labels collections of examples, which are called *bags*.

Multiple-Instance Problem. Formulation.

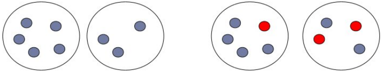
- In MIL, instead of giving the learner labels for the individual examples, the trainer only labels collections of examples, which are called *bags*.
- Each bag contains *many* instances.

Multiple-Instance Problem. Formulation.

- In MIL, instead of giving the learner labels for the individual examples, the trainer only labels collections of examples, which are called *bags*.
- Each bag contains *many* instances.
- A bag is labeled *negative* if all the instances in it are negative.

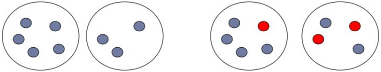
Multiple-Instance Problem. Formulation.

- In MIL, instead of giving the learner labels for the individual examples, the trainer only labels collections of examples, which are called *bags*.
- Each bag contains *many* instances.
- A bag is labeled *negative* if all the instances in it are negative.
- A bag is labeled *positive* if there is at least one instance in it which is positive.



Multiple-Instance Problem. Formulation.

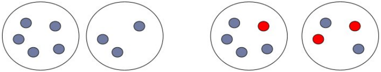
- In MIL, instead of giving the learner labels for the individual examples, the trainer only labels collections of examples, which are called *bags*.
- Each bag contains *many* instances.
- A bag is labeled *negative* if all the instances in it are negative.
- A bag is labeled *positive* if there is at least one instance in it which is positive.



- **Goal**: From a collection of labeled bags, the learner tries to induce a concept that will label individual bags correctly.

Multiple-Instance Problem. Formulation.

- In MIL, instead of giving the learner labels for the individual examples, the trainer only labels collections of examples, which are called *bags*.
- Each bag contains *many* instances.
- A bag is labeled *negative* if all the instances in it are negative.
- A bag is labeled *positive* if there is at least one instance in it which is positive.



- **Goal**: From a collection of labeled bags, the learner tries to induce a concept that will label individual bags correctly.
- The difficulty for learning this property is that it is unknown which of the instances is responsible for a positive classification of a bag.

AdaBoost Approaches

- Auer and Ortner, 2004,
- Blum, 1998,
- Anyboost by Mason,
- Our Approach

Auer and Ortner, 2004

- regular adaboost alg. with a weak learning algorithm that handles MIL.
- Uses bags weights.
- Weak hypotheses are balls of arbitrary center and radius with respect to some metric.
- Distribution accuracy: $D(h,w) = \sum_{h(B)=y(B)} w_B$ (quality of weak hypothesis)
- Demand $D(h,w)$ is $> 1/2 + \epsilon$, for $\epsilon > 0$.

Lemma

For each weight distribution $w = (w_{B_1}, \ldots, w_{B_{|B|}})$ there is a ball $h = h(x,r)$ in H s.t. $D(h,w) > 1/2 + \frac{1}{4k+2}$, where k is the number of positive bags in B.

Auer and Ortner, 2004

Brief idea

- For each instance x in the positive bag they compute a ball with center x and optimal radius r_0:

$$r_0 = max\{r' \geq 0 | D(h(x, r'), w) = max_r D(h(x, r), w)\}.$$

- To speed up the bags are sorted by distance to x.
- All instances inside the ball become positive and negative outside the ball.

Blum, 1998

- Classification algorithms are robust to data with noise: some instances can be mislabeled.

Blum, 1998

- Classification algorithms are robust to data with noise: some instances can be mislabeled.
- In the standard PAC-learning model, a learning algorithm is repeatedly given labeled examples of an unknown target concept, drawn independently from some probability distribution.

Blum, 1998

- Classification algorithms are robust to data with noise: some instances can be mislabeled.
- In the standard PAC-learning model, a learning algorithm is repeatedly given labeled examples of an unknown target concept, drawn independently from some probability distribution.
- Goal: approximate the target concept with respect to this distribution.

Blum, 1998

- Classification algorithms are robust to data with noise: some instances can be mislabeled.
- In the standard PAC-learning model, a learning algorithm is repeatedly given labeled examples of an unknown target concept, drawn independently from some probability distribution.
- Goal: approximate the target concept with respect to this distribution.
- Idea: treat ALL instances in the positive bag as positive instances: i.e. positive class noise and ignore the bag level information.

Blum, 1998

- Classification algorithms are robust to data with noise: some instances can be mislabeled.
- In the standard PAC-learning model, a learning algorithm is repeatedly given labeled examples of an unknown target concept, drawn independently from some probability distribution.
- Goal: approximate the target concept with respect to this distribution.
- Idea: treat ALL instances in the positive bag as positive instances: i.e. positive class noise and ignore the bag level information.
- There is a reduction to learning with one-sided or two-sided random classification noise.

Blum, 1998

- Classification algorithms are robust to data with noise: some instances can be mislabeled.
- In the standard PAC-learning model, a learning algorithm is repeatedly given labeled examples of an unknown target concept, drawn independently from some probability distribution.
- Goal: approximate the target concept with respect to this distribution.
- Idea: treat ALL instances in the positive bag as positive instances: i.e. positive class noise and ignore the bag level information.
- There is a reduction to learning with one-sided or two-sided random classification noise.
- Thus standard and well-developed algorithms can be used.

Anyboost, Mason et al

- *Gradient descent* alg. for choosing linear combination of elements of an inner product function space so as to minimize some cost function.

Anyboost, Mason et al

- *Gradient descent* alg. for choosing linear combination of elements of an inner product function space so as to minimize some cost function.
- Let (x, y) examples from $X \times Y$, interested in voted combination of classifiers of the form $sgn(F(x))$, where

$$F(x) = \sum_{t-1}^{T} \alpha_t h_t(x),$$

  h_t are base classifiers and α_t are classifiers weight.
- Margin of (x, y) is wrt $sgn(F(x))$ is $yF(x)$
- **Goal**: $P(sgn(F(x)) \neq y)$ is small.

Anyboost, Mason et al

- Find voted classifiers which minimize the sampling average of some cost function of the margin.
- Thus need to find F such that

$$C(F) = \frac{1}{m} \sum_{i=1}^{m} C(y_i F(x_i))$$

is minimized for some suitable cost function $C : R \to R$

- Using gradient decent method outputs $f \in \mathcal{F}$ with a large value of $- < \nabla C(F), f >$

Table 1: Existing voting methods viewed as AnyBoost on margin cost functions.

Algorithm	Cost function	Step size
AdaBoost [9]	$e^{-yF(x)}$	Line search
ARC-X4 [2]	$(1 - yF(x))^5$	$1/t$
ConfidenceBoost [19]	$e^{-yF(x)}$	Line search
LogitBoost [12]	$\ln(1 + e^{-yF(x)})$	Newton-Raphson

AnyBoost MIL cost function: Viola, 2006

- Noisy-OR Boost: They have custom cost function which weights on the instance level
- Use weak classifier uses these weights (do not need weak MIL learner)
- Thus MIL is handles by a proposed cost function and not by the weak learning algorithm.
- In order to make weak classification more accurate they down-weight instances in the pos. bag believed to be negative.
- Boosting utilizes MIL information through cost function and weights.

Our Idea. Bag-level prediction

- We followed the original paper (Schapire, 1997) and proposed the following approach for bag-level prediction.
- Weak classifier gives us not only the label of each instance but also the **probability** of an instance being positive.
- We incorporated this knowledge and created a weak hypothesis for a bag using weak hypothesis for the instances.

Our version of Boosting Algorithm for MIL

- Given (x_i, y_i), i: bags, each bag has different number of instances

Our version of Boosting Algorithm for MIL

- Given (x_i, y_i), i: bags, each bag has different number of instances
- $w_{1i} = 1/N$

Our version of Boosting Algorithm for MIL

- Given (x_i, y_i), i: bags, each bag has different number of instances
- $w_{1i} = 1/N$
- Weak hyp. for bag using weak hypothesis for the instances

$$H_t(x_i) = \frac{\sum_j (P_{tij} \cdot h_{tij})}{\sum_j P_{tij}}$$

- error: $e_t = \sum_i w_{ti} H_t(x_i) y_i$

Our version of Boosting Algorithm for MIL

- Given (x_i, y_i), i: bags, each bag has different number of instances
- $w_{1i} = 1/N$
- Weak hyp. for bag using weak hypothesis for the instances

$$H_t(x_i) = \frac{\sum_j (P_{tij} \cdot h_{tij})}{\sum_j P_{tij}}$$

- error: $e_t = \sum_i w_{ti} H_t(x_i) y_i$
- Choose α_t

Our version of Boosting Algorithm for MIL

- Given (x_i, y_i), i: bags, each bag has different number of instances
- $w_{1i} = 1/N$
- Weak hyp. for bag using weak hypothesis for the instances

$$H_t(x_i) = \frac{\sum_j (P_{tij} \cdot h_{tij})}{\sum_j P_{tij}}$$

- error: $e_t = \sum_i w_{ti} H_t(x_i) y_i$
- Choose α_t
- $w_{t+1,i} = \frac{w_{ti} exp(-\alpha_t H_t(x_i) y_i)}{Z_t}$

Our version of Boosting Algorithm for MIL

- Given (x_i, y_i), i: bags, each bag has different number of instances
- $w_{1i} = 1/N$
- Weak hyp. for bag using weak hypothesis for the instances

$$H_t(x_i) = \frac{\sum_j (P_{tij} \cdot h_{tij})}{\sum_j P_{tij}}$$

- error: $e_t = \sum_i w_{ti} H_t(x_i) y_i$
- Choose α_t
- $w_{t+1,i} = \frac{w_{ti} exp(-\alpha_t H_t(x_i) y_i)}{Z_t}$
- Final output: $H(x) = sign(\sum_t \alpha_t H_t(x))$.

Quick Proof

- α_t is the key value. The question is HOW to choose the learning parameter α.

First we prove an easy theorem:

Theorem

The following error holds on the training error of H:

$$\frac{1}{N}|\{i : H(x_i) \neq y_i\}| \leq \prod_{t=1}^{T} Z_t,$$

where N is a number of bags and Z_t - normalization factor.

Proof

Let $f(x_i) = \sum_t \alpha_t H_t(x_i)$, then

$$w_{T+1}(i) = \frac{exp(-y_i \sum_t \alpha_t H_t(x_i))}{N \prod_t Z_t}$$

Proof

Let $f(x_i) = \sum_t \alpha_t H_t(x_i)$, then

$$w_{T+1}(i) = \frac{exp(-y_i \sum_t \alpha_t H_t(x_i))}{N \prod_t Z_t} = \frac{exp(-y_i f(x_i))}{N \prod_t Z_t}$$

Also if $H(x_i) \neq y_i$ then $y_i f(x_i) \leq 1$ thus $exp(-y_i f(x_i)) \geq 1$.
Then predicate $[H(x_i) \neq y_i] \leq exp(-y_i f(x_i))$.
Finally:

$$\begin{aligned} & \frac{1}{N} \sum_i [H(x_i) \neq y_i] \leq \frac{1}{N} \sum_i exp(-y_i f(x_i)) = \\ = \quad & \sum_i \left(\prod_t Z_t\right) w_{T+1}(i) = \prod_t Z_t. \end{aligned} \tag{1}$$

Choosing α_t

- Let $u_i = y_i H_t(x_i)$

$$Z = \sum_i w(i) exp(-\alpha u_i)$$

Choosing α_t

- Let $u_i = y_i H_t(x_i)$

$$Z = \sum_i w(i) exp(-\alpha u_i) \leq \sum_i w(i) \left(\frac{1 + u_i}{1} e^{-\alpha} + \frac{1 - u_i}{2} e^{\alpha} \right)$$

Choosing α_t

- Let $u_i = y_i H_t(x_i)$

$$Z = \sum_i w(i) exp(-\alpha u_i) \leq \sum_i w(i) \left(\frac{1+u_i}{1} e^{-\alpha} + \frac{1-u_i}{2} e^{\alpha} \right)$$

- Need to choose α to minimize the right hand side.
- $\alpha = \frac{1}{2} \ln(\frac{1+err}{1-err})$, where $err = \sum_i w_i H(x_i) y_i$

Receiver **O**perating **C**haracteristic - ROC

– is a graphical plot of *true positive* rate, vs. *false positive* rate for a binary classifier systems

		actual value		
		p	***n***	**total**
prediction outcome	***p'***	True Positive	False Positive	P'
	n'	False Negative	True Negative	N'
	total	P	N	

ROC curve

A ROC space is defined by FPR and TPR as x and y axes respectively, which depicts relative trade-offs between true positive (benefits) and false positive (costs).

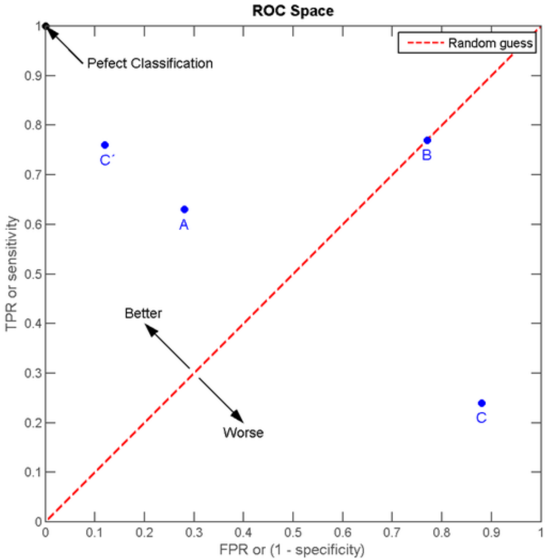
A

TP=63	FP=28	91
FN=37	TN=72	109
100	100	200

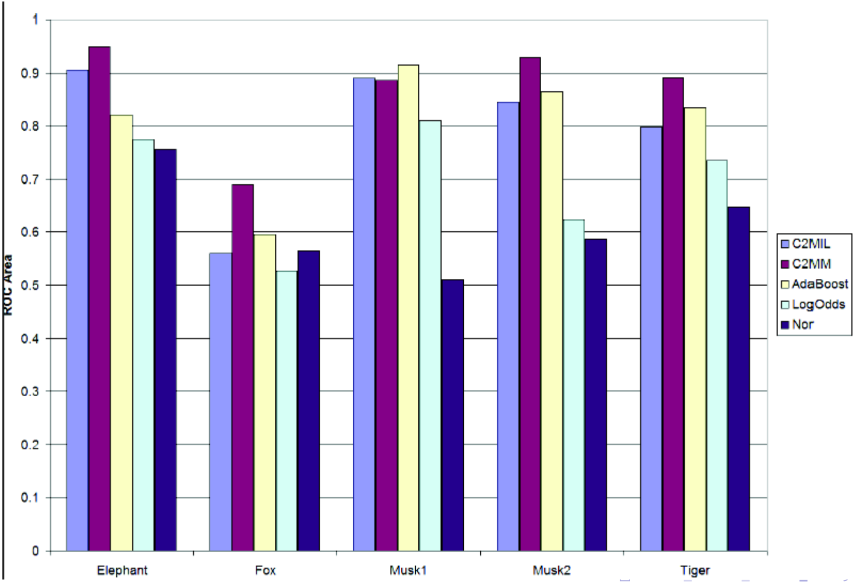
TPR = 0.63

FPR = 0.28

ROC curve


ROC Space
Random guess
Pefect Classification
C´
A
B
C
Better
Worse
TPR or sensitivity
FPR or (1 - specificity)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

Results



ROC Area
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Elephant
Fox
Musk1
Musk2
Tiger
C2MIL
C2MM
AdaBoost
LogOdds
Nor

Difference Between Algorithms

- Auer (Hyper-Balls)
  - Weights on bags versus both bags and instances
  - Weak learner: MIL vs. classifier
- Blum (AdaBoost Classifer)
  - Ours utilizes bag information
- Viola (AnyBoost with Noisy-OR)
  - AnyBoost versus AdaBoost (choice of α and weight update)
  - Cost function: Noisy-OR versus Asymmetric-OR

Open Problems

- Re-weighing a bag, best distribution and to prove something about it
- Another cost function that we can prove theoretical results for.

Derivative approach (Schapire et al)

$Z(\alpha) = Z = \sum_i w(i) exp(-\alpha u_i)$. The first derivative is

$$Z'(\alpha) = -\sum_i w(i) u_i exp(-\alpha u_i) = -Z \sum_i w_{t+1}(i) u_i$$

Thus, if w_{t+1} is formed using the value of α_t which minimizes Z_t, we'll have

$$\sum_i w_{t+1}(i) u_i = 0.$$

We can numerically find the unique minimum of $Z(\alpha)$ by a simple binary search, or more sophisticated numerical methods.

Another problem

- Maybe there is a "better" upper bound for $Z(\alpha)$ that allows to find a closed form solution.