



A Survey on Boosting Algorithms for Supervised Learning

Artur Ferreira

supervisor Prof. Mário Figueiredo

26 November 2007



Summary

1. Ensembles of classifiers and boosting
2. Boosting: origins and features
3. The AdaBoost algorithm and its variants
 - Supervised Learning (SL)
 - Semi-Supervised Learning (SSL)
4. Experimental results (binary classification)
 - Comparison with other techniques
5. Concluding remarks (and ongoing work)

1. Ensembles of classifiers

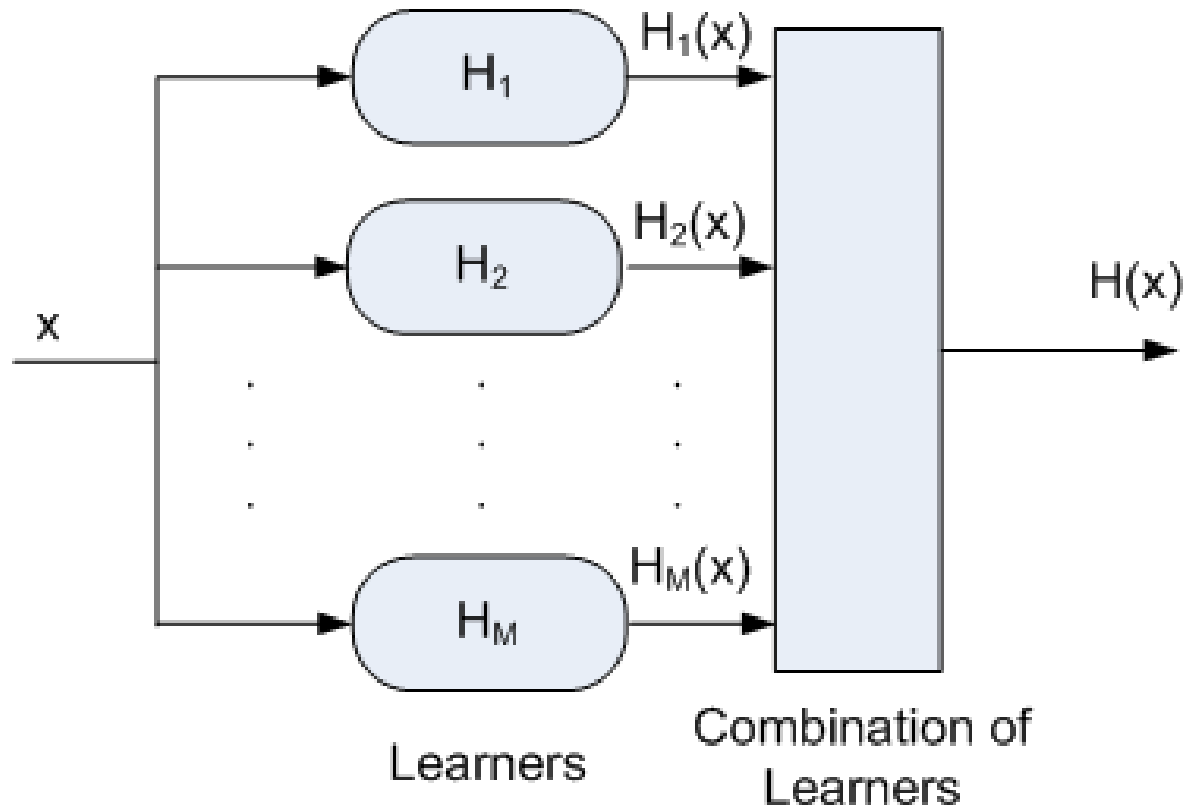
■ Motivation for ensembles of classifiers

- Instead of learning a single complex classifier, learn several simple classifiers
- Combine the output of the simple classifiers to produce the classification decision

- For instance, instead of training a large neural network (NN), train several smaller NN and combine their individual output in order to produce the final output

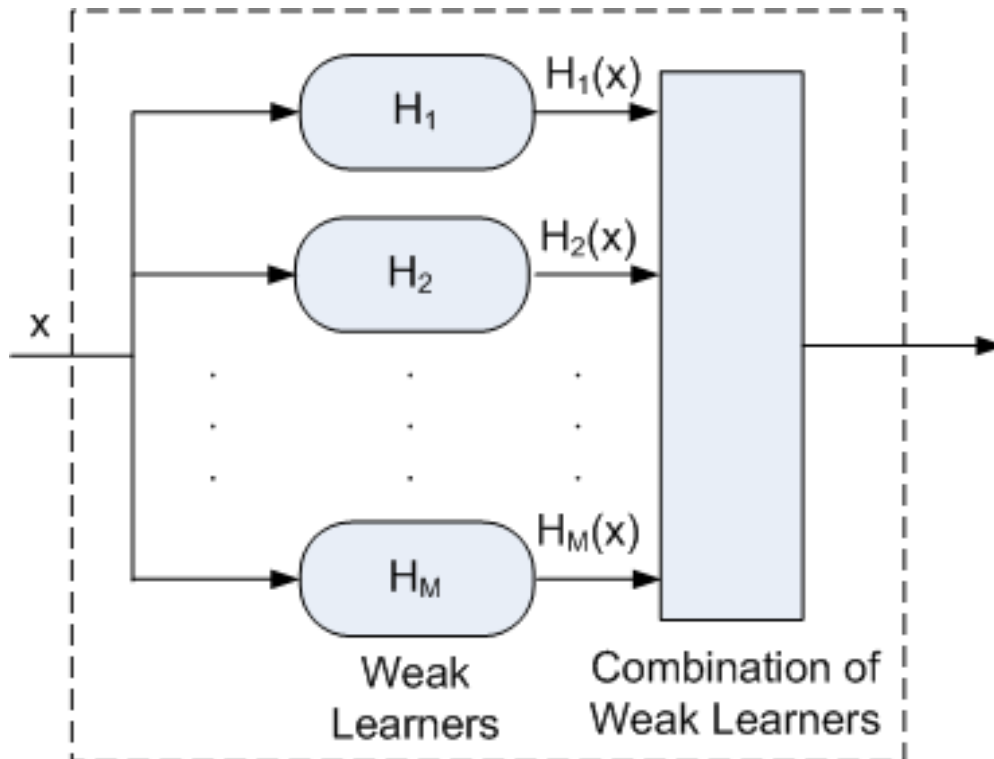
1. Ensembles of classifiers

- $H_1(x), \dots, H_M(x)$ are the weak learners
- The output classification decision is given by $H(x)$



2. Boosting: origins and features

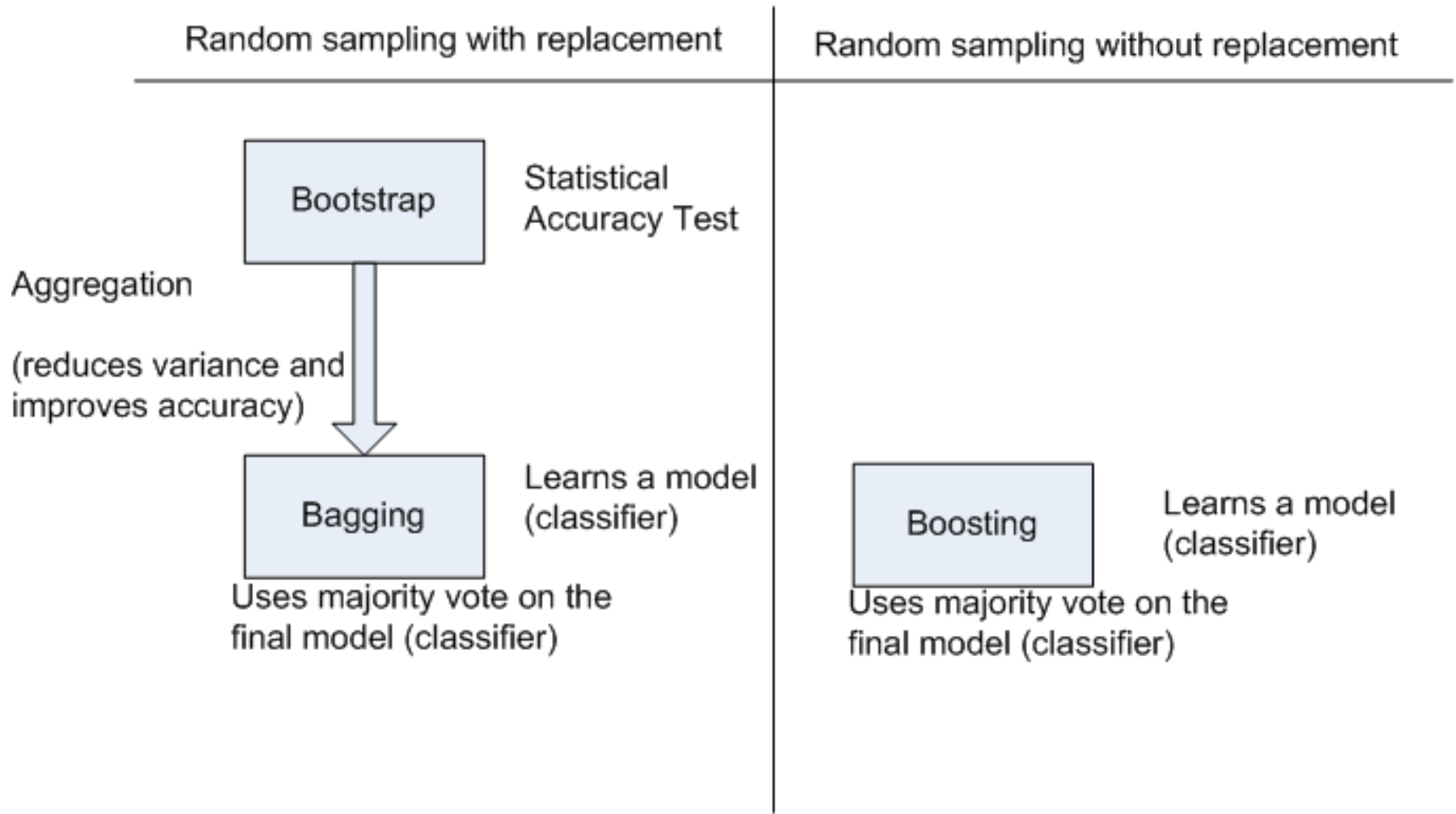
- Boosting builds an ensemble of classifiers
- Combine simple classifiers to build a robust classifier
- Each classifier H_m has an associated contribution α_m



$$H(\mathbf{x}) = \text{sign} \left(\sum_{m=1}^M \alpha_m H_m(\mathbf{x}) \right)$$

2. The origins of Boosting

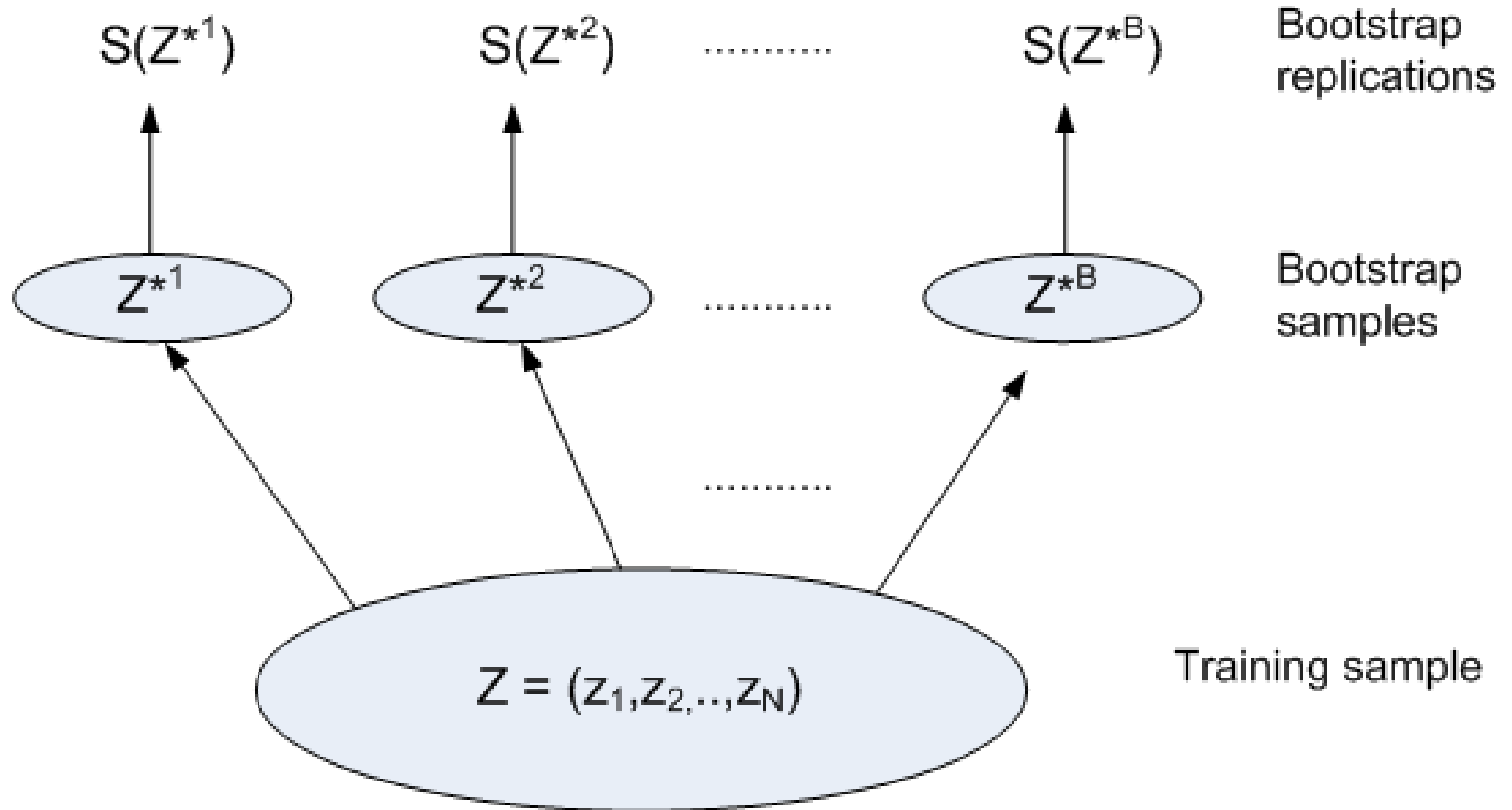
- Boosting is related to other techniques such as Bootstrap and Bagging



2. Bootstrap: features

- General purpose sample-based statistical method
- Assesses the statistical accuracy of some estimate $S(Z)$, over a training set $Z = \{z_1, z_2, \dots, z_N\}$, with $z_i = (x_i, y_i)$
- Draws randomly with replacement from a set of data points,
- Uses B versions of the training set
- To check the accuracy, the measure is applied over the B sampled versions of the training set

2. Bootstrap: graphical idea



2. Bootstrap: algorithm

Input:

- Training set $Z = \{z_1, z_2, \dots, z_N\}$, with $z_i = (x_i, y_i)$.
- B , number of sampled versions of the training set.

Output:

- $S(Z)$, statistical estimate and its accuracy.

Step 1

for $n=1$ to B

a) Draw, with replacement, $L < N$ samples from the training set Z , obtaining the n th sample Z^{*n} .

b) For each sample Z^{*n} , estimate a statistic $S(Z^{*n})$.

end

Step 2

Produce the bootstrap estimate $S(Z)$, using $S(Z^{*n})$ with $n = \{1, \dots, B\}$.

Step 3

Compute the accuracy of the estimate, using the variance or some other criterion.

2. Bagging: features

- Proposed by Breiman, 1996
- Consists of Bootstrap aggregation
- Training set $Z = \{z_1, z_2, \dots, z_N\}$, with $z_i = (x_i, y_i)$ for which we intend to fit a model $f(x)$
- Obtain a prediction of $f(x)$ at input x
- For each bootstrap sample b we fit our model, giving a prediction $f_b(x)$.
- The Bagging estimate is the average of the individual estimates

2. Bagging: algorithm (classification)

Input:

- Training set $Z=\{z_1, z_2, \dots, z_N\}$, with $z_i=(x_i, y_i)$.
- B , number of sampled versions of the training set.

Output:

- $H(x)$, a classifier suited for the training set.

Step 1

for $n=1$ to B

a) Draw, with replacement, $L < N$ samples from the training set Z , obtaining the n th sample Z^{*n} .

b) For each sample Z^{*n} , learn classifier H_n .

end

Step 2

Produce the final classifier as a vote of H_n with $n=\{1, \dots, B\}$.

$$H(x) = \text{sgn} \left(\sum_{n=1}^B H_n(x) \right)$$

2. Boosting: features

- Freund and Schapire, 1989
- Similar to Bagging, in the sense that uses several versions of the training set
- Uses 3 subsets of the training set
- Differs from Bagging, in the sense that does not allow replacement
- The final classification is done by a majority vote (3 classifiers)

2. Boosting: algorithm (classification)

Input:

- Training set $Z = \{z_1, z_2, \dots, z_N\}$, with $z_i = (x_i, y_i)$.

Output:

- $H(x)$, a classifier suited for the training set.

Step 1 - Draw, without replacement, $L < N$ samples from the training set Z , obtaining Z^{*1} ; train weak learner H_1 on Z^{*1}

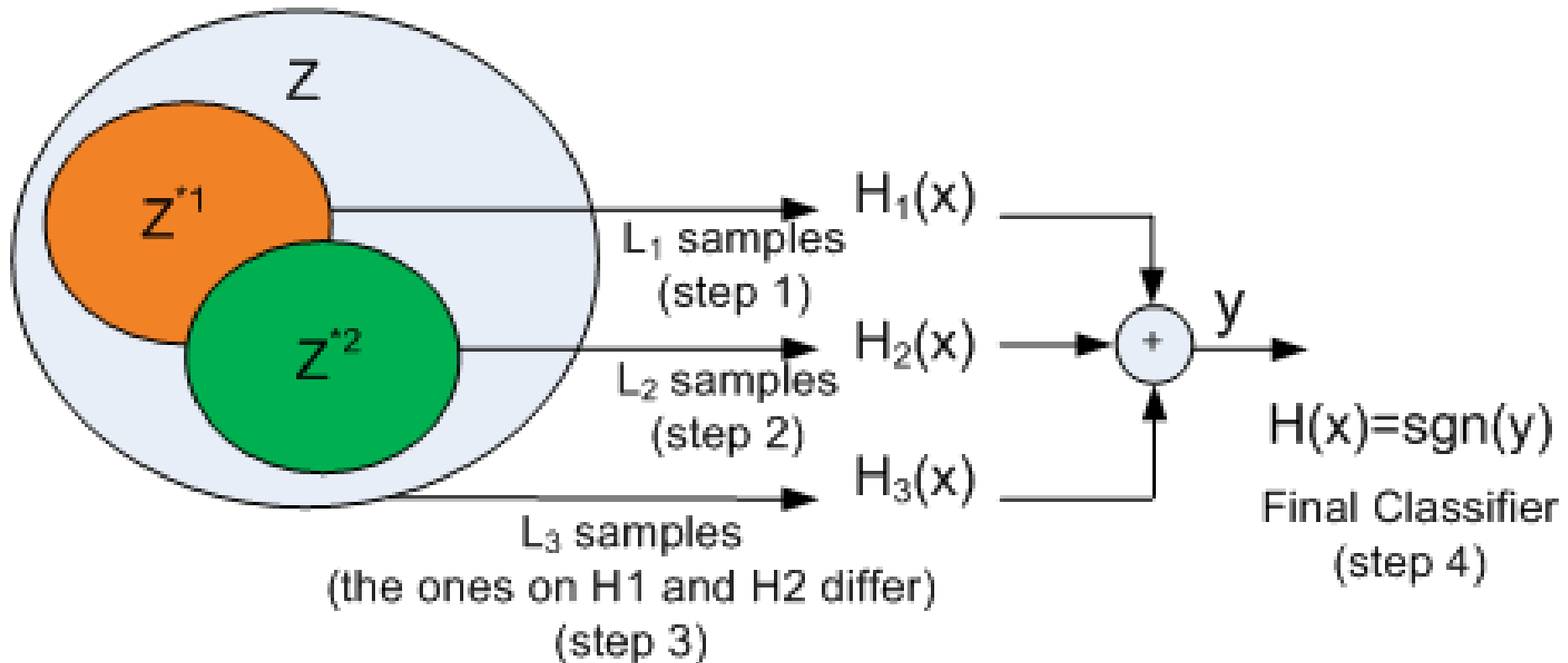
Step 2 - Select $L_2 < N$ samples from Z with half of the samples misclassified by H_1 to obtain Z^{*2} ; train weak learner H_2 on it.

Step 3 - Select all samples from Z that H_1 and H_2 disagree on; train weak learner H_3 , using these samples.

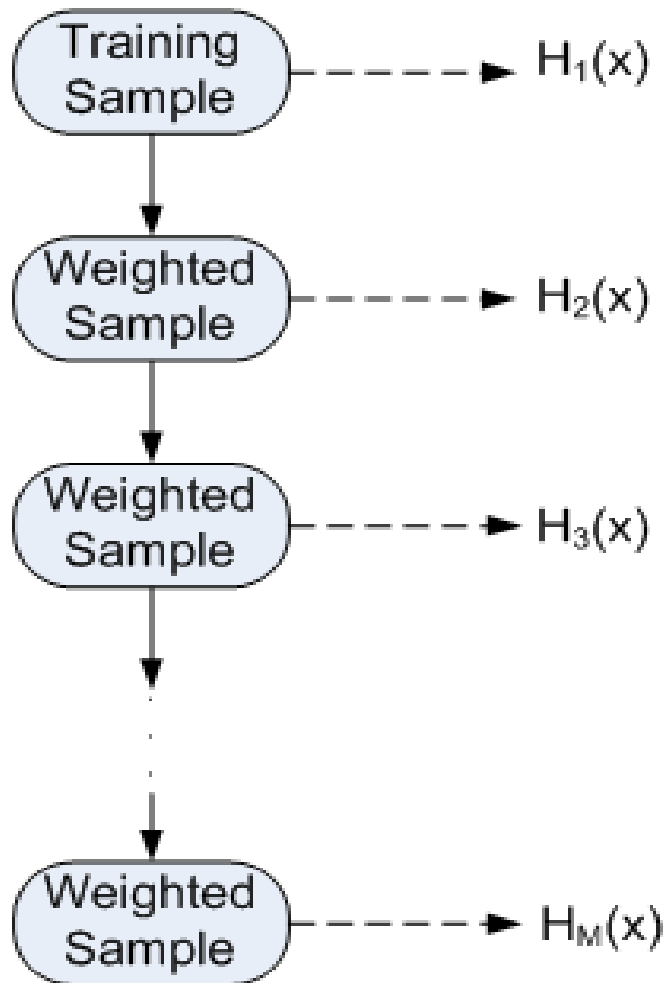
Step 4 - Produce the final classifier as a vote of the three weak learners

$$H(x) = \text{sgn} \left(\sum_{n=1}^3 H_n(x) \right)$$

2. Boosting: graphical idea

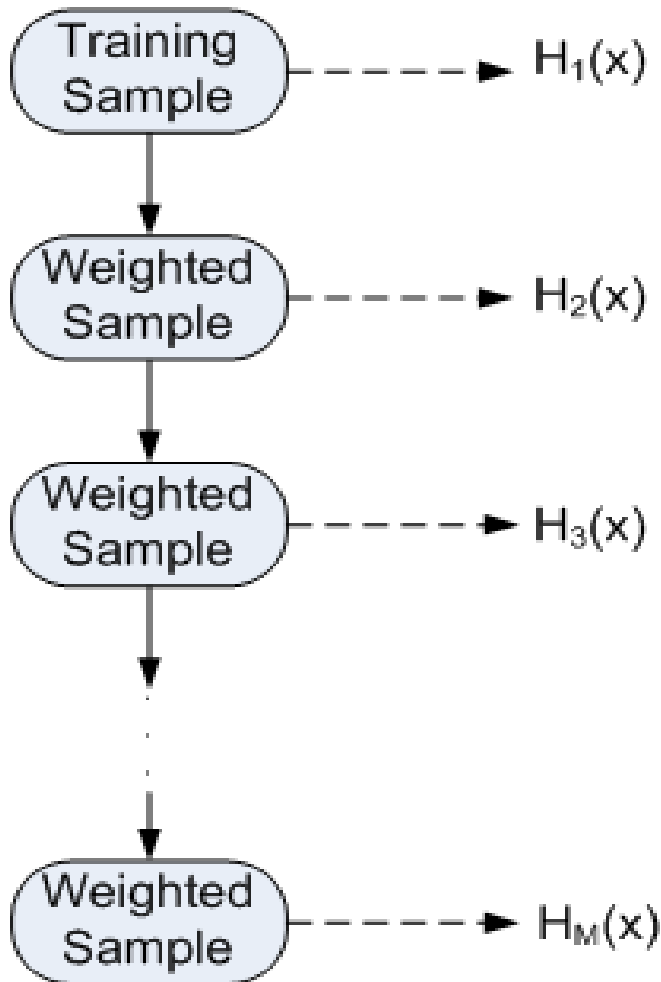


3. AdaBoost Algorithm



- Adaptive Boosting
- By Freund and Schapire, 1996
- Instead of sample, reweight the data
- Consecutive classifiers are learned on weighed versions of the data
- The entire training set is considered in order to learn each classifier

3. AdaBoost Algorithm



- Consecutive classifiers are learned on weighed versions of the data
- The weight is given by the (mis)classification of the previous classifiers
- The next classifier focus on the most difficult patterns
- The algorithm stops when:
 - the error rate of a classifier is >0.5
 - it reaches M classifiers

3. AdaBoost Algorithm

AdaBoost

Input: Training set (x,y)

N = # input patterns

M = # classifiers

Output: The ensemble given by

$$H(\mathbf{x}) = \text{sign} \left(\sum_{m=1}^M \alpha_m H_m(\mathbf{x}) \right)$$

1. Initialize the weights w_i , with $w_i = \frac{1}{N}$ $i \in \{1, \dots, N\}$
2. For $m=1$ to M
 - a) Fit a classifier H_m to data using weights w_i
 - b) Compute the error rate of the classifier err_m
 - c) Compute the contribution $\alpha_m = 0.5 \log \left(\frac{1-\text{err}_m}{\text{err}_m} \right)$
 - d) Update the weights $w_i \leftarrow w_i \exp(-\alpha_m I(y_i \neq H_m(\mathbf{x})))$

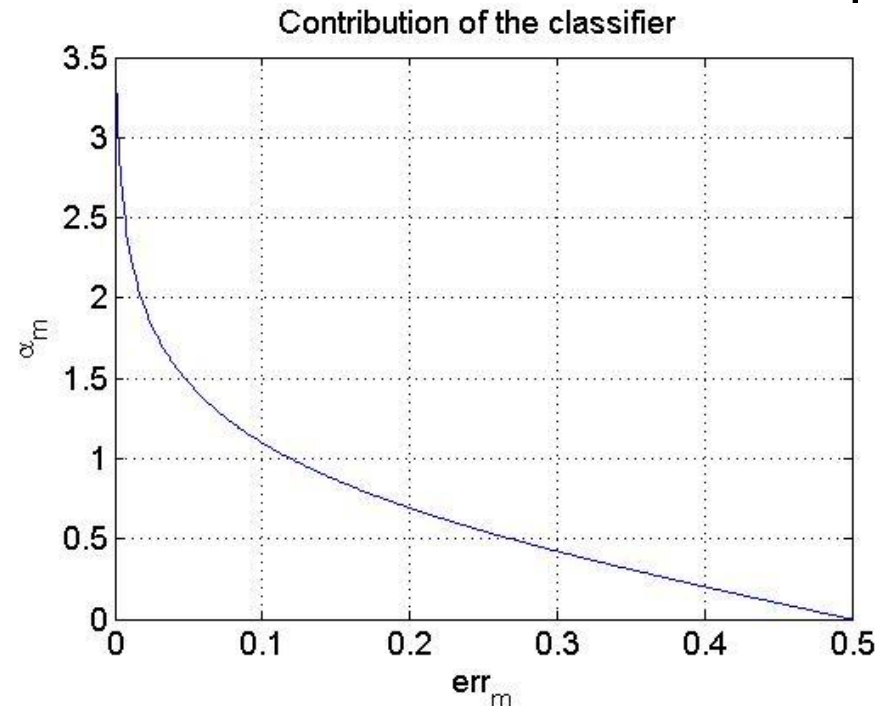
3. Output the ensemble

$$H(\mathbf{x}) = \text{sign} \left(\sum_{m=1}^M \alpha_m H_m(\mathbf{x}) \right)$$

3. AdaBoost Algorithm: detail 1


1. Initialize the weights
2. For $m=1$ to M
 - a) Fit a classifier to data using current weights
 - b) Compute the error rate of the current classifier
 - c) Compute the weighted contribution of the classifier
 - d) Update the weight of each input pattern
3. Output the ensemble

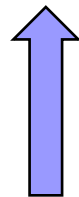
$$\alpha_m = 0.5 \log \left(\frac{1 - \text{err}_m}{\text{err}_m} \right)$$



3. AdaBoost Algorithm: detail 2

1. Initialize the weights
2. For $m=1$ to M
 - a) Fit a classifier to data using current weights
 - b) Compute the error rate of the current classifier
 - c) Compute the weighted contribution of the classifier
 - d) Update the weight of each input pattern
3. Output the ensemble


$$w_i \leftarrow w_i \exp(-\alpha_m I(y_i \neq H_m(\mathbf{x})))$$



Indicator function $\begin{cases} -1, & \text{different} \\ 1, & \text{equal} \end{cases}$

\longrightarrow Weight increases

\longrightarrow Weight decreases

3. AdaBoost Algorithm

- Comparison of Boosting (1989) and AdaBoost (1996)

Feature	Boosting	AdaBoost
Data processing	Random sampling without replacement	Weighting (no sampling)
Num. of classifiers	Three	Up to M
Decision	Majority vote	Weighted vote

3. AdaBoost Algorithm: variants

- Variants for SL

Boosting (1989)
AdaBoost (1996)

- Binary case

- Since 1999, several variants have been proposed

Real AdaBoost (1999)
Margin Boost (2000)
Modest AdaBoost (2000)
Gentle AdaBoost (2000)
AnyBoost (2000)
MarginBoost (2000)
KLBoost (2003)
FloatBoost (2004)
ActiveBoost (2004)
JensenShannon Boost (2005)
Infomax Boost (2005)
Emphasis Boost (2006)
Entropy Boost (2007)
Reweight Boost (2007)
LogitBoost (????)
Brown Boost (????)
Weight Boost (????)

3. AdaBoost Algorithm: variants

- Variants for SL
- Multiclass case
- Since 1997, several variants have been proposed

Boosting (1989)
AdaBoost (1996)

AdaBoost.M1 (1997)
AdaBoost.M2 (1997)
AdaBoost.OC (1997)
AdaBoost.ECC (1999)
AdaBoost.M1W (2002)
BoostMA (2005)
AdaBoost.ERP (2006)

3. AdaBoost Algorithm: variants

- Variants for SSL
- Since 2001, only three variants have been proposed
- The key issue is how to handle unlabeled data
 - usually there is some loss (or contrast) function
 - this my current topic of research and maybe the title of my next talk...!

Boosting (1989)

AdaBoost (1996)

MixtBoost (2001)

SSMarginBoost (2002)

SemiBoost (2007)

3. AdaBoost variants

■ Variants for SL to consider:

- Real AdaBoost (1999)
- Gentle Boost (2000)
- Modest AdaBoost (2005)
- Reweight Boost (2007)

1. Initialize the weights
2. For $m=1$ to M
 - a) Fit a classifier to data using current weights
 - b) Compute the error rate of the current classifier
 - c) Compute the weighted contribution of the classifier
 - d) Update the weight of each input pattern
3. Output the ensemble



Usually, variants change only these steps

3. AdaBoost variants: Real AdaBoost

Input:

- Training set $Z=\{z_1, z_2, \dots, z_N\}$, with $z_i=(x_i, y_i)$.
- M , the maximum number of classifiers.

Output:

- $H(x)$, a classifier suited for the training set.

Step 1

Initialize the weights $w_i = 1/N$, $i=\{1, \dots, N\}$

Step 2

for $m=1$ to M

a) Fit the class probability estimate $p_m(x)=P_w(y=1|x)$, using weights w_i on the training data.

b) Set $H_m(x)=0.5 \log \left(\frac{1-p_m(x)}{p_m(x)} \right)$

c) Set weights $w_i \leftarrow w_i \exp \left(\epsilon y_i H_m(x) \right)$ and renormalize.

Step 3

Produce the final classifier. $H(x) = \text{sgn} \left(\sum_{n=1}^M H_n(x) \right)$

AdaBoost variants

- Real AdaBoost (RA) differs from AdaBoost on steps 2a) and 2b)
 - On RA, these steps consist on the calculation of the probability that a given pattern belongs to a class.
 - The AdaBoost algorithm classifies the input patterns and calculates the weighted amount of error.
- RA performs exact optimization with respect to H_m
- Gentle AdaBoost (GA) improves it, using weighted least-squares regression
- Usually, GA produces a more reliable and stable ensemble.

3. AdaBoost variants: Gentle AdaBoost

Input:

- Training set $Z = \{z_1, z_2, \dots, z_N\}$, with $z_i = (x_i, y_i)$.
- M , the maximum number of classifiers.

Output:

- $H(x)$, a classifier suited for the training set.

Step 1

Initialize the weights $w_i = 1/N$, $i = \{1, \dots, N\}$

Step 2

for $m=1$ to M

- a) Train $H_m(x)$ by weighted least squares of x_i to y_i , with weights w_i
- b) Set $H(x) = H(x) + H_m(x)$
- c) Set weights $w_i \leftarrow w_i \exp(-y_i H_m(x))$ and renormalize.

Step 3

Produce the final classifier.

$$H(x) = \text{sgn} \left(\sum_{n=1}^M H_n(x) \right)$$

AdaBoost variants

- Modest AdaBoost (MA) improves Gentle AdaBoost (GA) improves it, using an “inverted” distribution

$$\bar{w} = 1 - w$$

- this distribution gives higher weights to samples that are already correctly classified by earlier classifiers
- Usually MA attains ensembles with less generalization error than GA
- But, MA typically has higher training error than GA

3. AdaBoost variants: Modest AdaBoost

Input:....

Output:... The same as before

Step 1 - Initialize the weights $w_i = 1/N, i=\{1,\dots,N\}$

Step 2

for $m=1$ to M

a) Train $H_m(x)$ by weighted least squares of x_i to y_i , with weights w_i

b) Compute inverted distribution $\bar{w} = 1 - w$ and renormalize.

c) Compute $P_m^{+1} = P_w(y = +1, H_m(x))$

$$P_m^{-1} = P_w(y = -1, H_m(x))$$

$$\bar{P}_m^{+1} = P_{\bar{w}}(y = +1, H_m(x))$$

$$\bar{P}_m^{-1} = P_{\bar{w}}(y = -1, H_m(x))$$

d) Set $H_m(x) = P_m^{+1}(1 - \bar{P}_m^{+1}) - P_m^{-1}(1 - \bar{P}_m^{-1})$

e) Update $w_i \leftarrow w_i \exp \left(\left\langle y_i H_m(x) \right\rangle \right)$

How good the current classifier is at predicting class labels.

How well the current classifier is on the data previously classified by earlier classifiers.

Step 3

Produce the final classifier.

$$H(x) = \text{sgn} \left(\sum_{n=1}^M H_n(x) \right)$$

3. AdaBoost variants: Modest AdaBoost

- The update on step 2d) is

$$H_m(x) = P_m^{+1}(1 - \bar{P}_m^{+1}) - P_m^{-1}(1 - \bar{P}_m^{-1})$$

- Decreases the contribution of a given classifier if it works “too well” on the previously correctly classified data, with high margin
- Increases the contribution of the classifiers that have high certain on the current classification
- The algorithm is named “Modest” because the class tend to work in “their domain”

3. AdaBoost variants: Reweight Boost

Input:....

r (>0), the number of previous classifiers

Output:... The same as before

Step 1 - Initialize the weights $w_i = 1/N$, $i=\{1,\dots,N\}$

Step 2

for $m=1$ to M

a) Fit a classifier $H_m(x)$ to the data with weights w_i

b) **Get combined classifier $H_r(x)$ from the previous r classifiers**

c) Compute error

d) Compute the contribution

e) Update the weights

Step 3

Produce the final classifier.

4. Experimental results

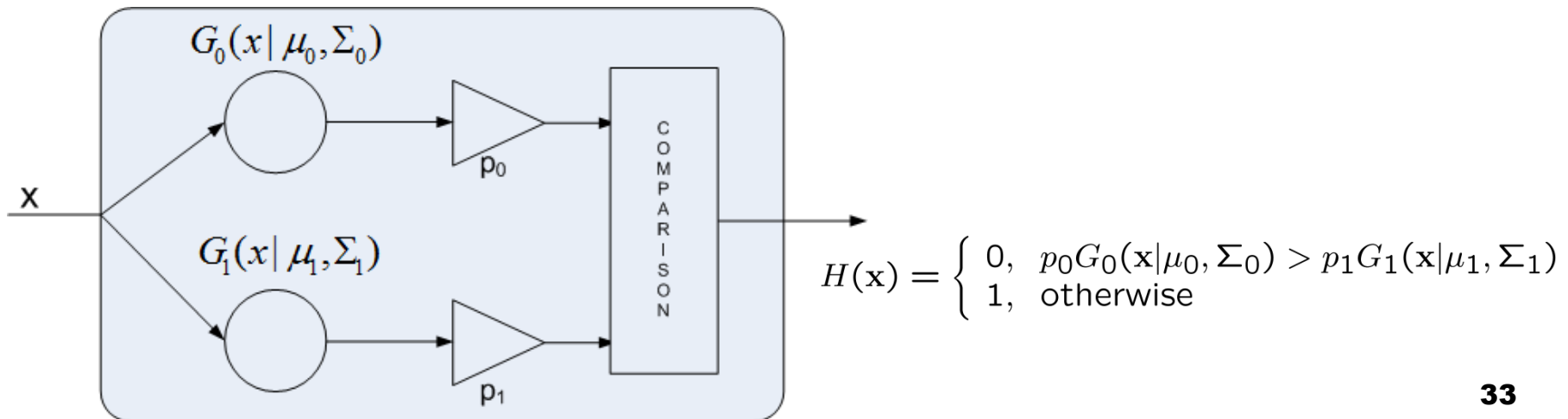
- Motivated by the claim
 - “AdaBoost with trees is the best off-the shelf classifier”
- and by this proven result
 - AdaBoost performs quite well, when each classifier is just a little better than random guessing (Friedam et al, 2001)
- We carried out some results on synthetic and real data using the weak classifiers:
 - Generative (2 gaussians)
 - RBF (Radial Basis Function) Unit with 2 gaussians and one output layer

4. Generative Weak Learner (AG)

- The generative weak learner has two multivariate Gaussian functions G_0 and G_1 (one per class)
 - p_0 and p_1 are the (sample) probabilities of each class
 - Gaussians learned by weighted mean and covariance estimation

$$\mu_j = \frac{\sum_{i=1}^N w_i x_i I(y_i = j)}{\sum_{i=1}^N w_i I(y_i = j)}$$

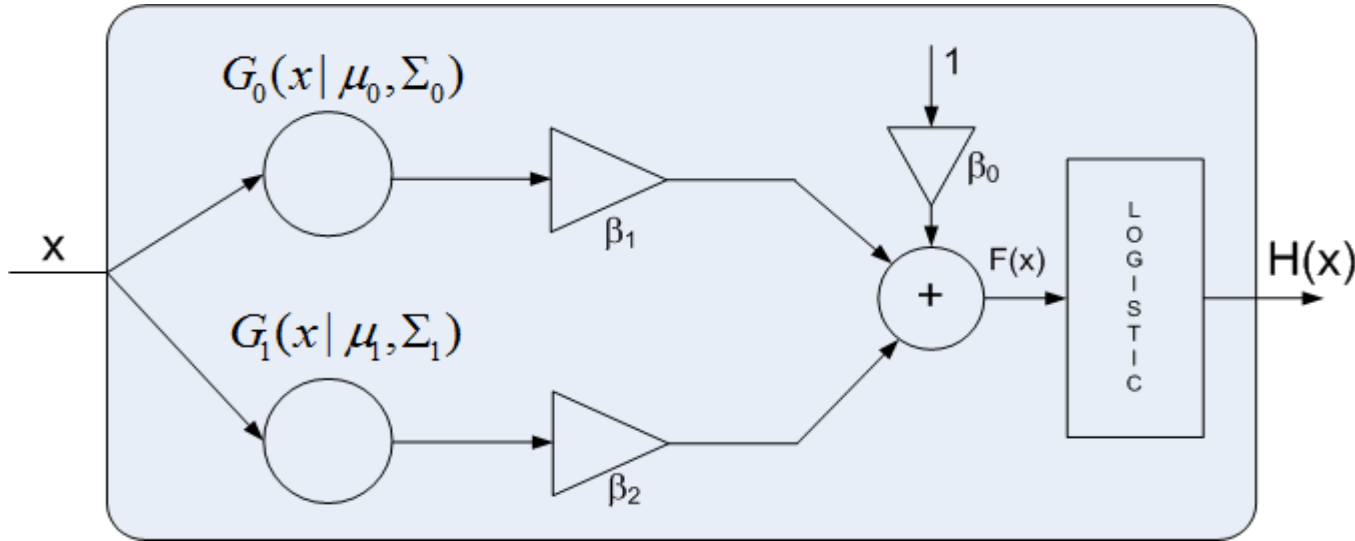
$$\Sigma_j = \frac{\sum_{i=1}^N w_i x_i x_i^T I(y_i = j)}{\sum_{i=1}^N w_i I(y_i = j)} - \mu_j \mu_j^T, \quad j \in \{0, 1\}$$



4. RBF Weak Learner (ARBF)

- The RBF (Radial Basis Function) weak learner has
 - two multivariate Gaussian functions (one per class)
 - one output layer
- The logistic function is applied on the output layer

$$H(\mathbf{x}) = \text{logistic}(F(\mathbf{x})) = \frac{1}{1 + \exp\left(-\beta_0 - \sum_{j=1}^2 \beta_j G_j(\|\mathbf{x} - \mu_j\|)\right)}$$



4. RBF Weak Learner (ARBF)

- Training of the RBF classifier is carried out by one-stage (global) algorithm*

- Mean and covariance matrix of each Gaussian are learned by the EM algorithm for mixture of Gaussians (**generative approach**)

$$z_i^s = \frac{\hat{\alpha}_s G(\mathbf{x}_i | \hat{\mu}_s, \hat{\Sigma}_s)}{\sum_{r=1}^k \hat{\alpha}_r G(\mathbf{x}_i | \hat{\mu}_r, \hat{\Sigma}_r)} \quad \hat{\alpha}_s = \frac{1}{N} \sum_{i=1}^N z_i^s \quad \left| \text{E-Step} \right.$$
$$\hat{\mu}_s = \frac{\sum_{i=1}^N \mathbf{x}_i z_i^s}{\sum_{i=1}^N z_i^s}, \quad \hat{\Sigma}_s = \frac{\sum_{i=1}^N (\mathbf{x}_i - \hat{\mu}_s)(\mathbf{x}_i - \hat{\mu}_s)^T z_i^s}{\sum_{i=1}^N z_i^s} \quad \left| \text{M-Step} \right.$$

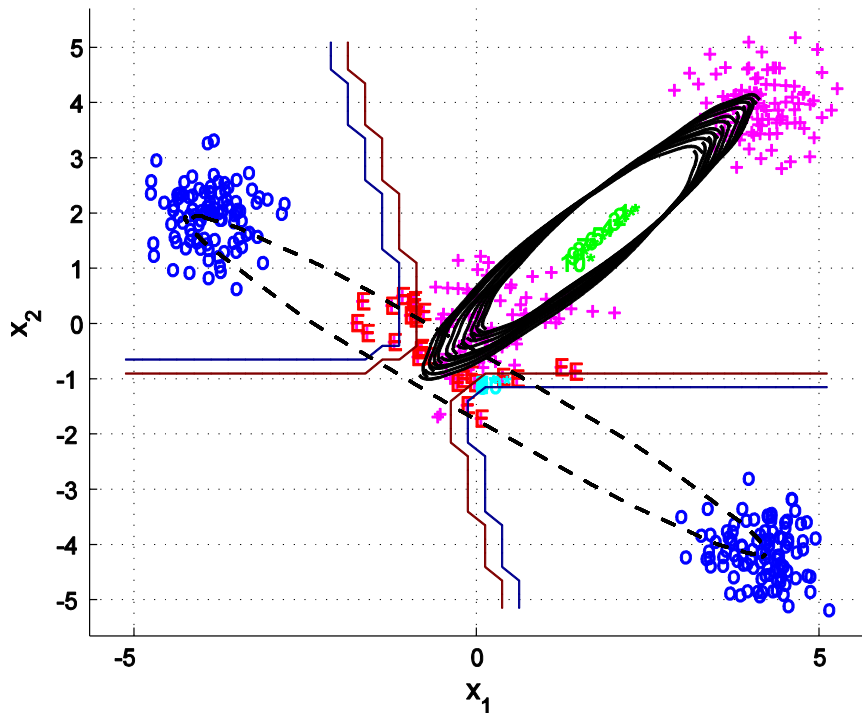
- Simultaneously, the output layer coefficients are learned by logistic regression (**discriminative approach**), by the Newton-Raphson step

$$\beta \leftarrow \beta + (\mathbf{X}^T \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^T (\mathbf{y} - \mathbf{p})$$

4. Experimental results – synthetic 1

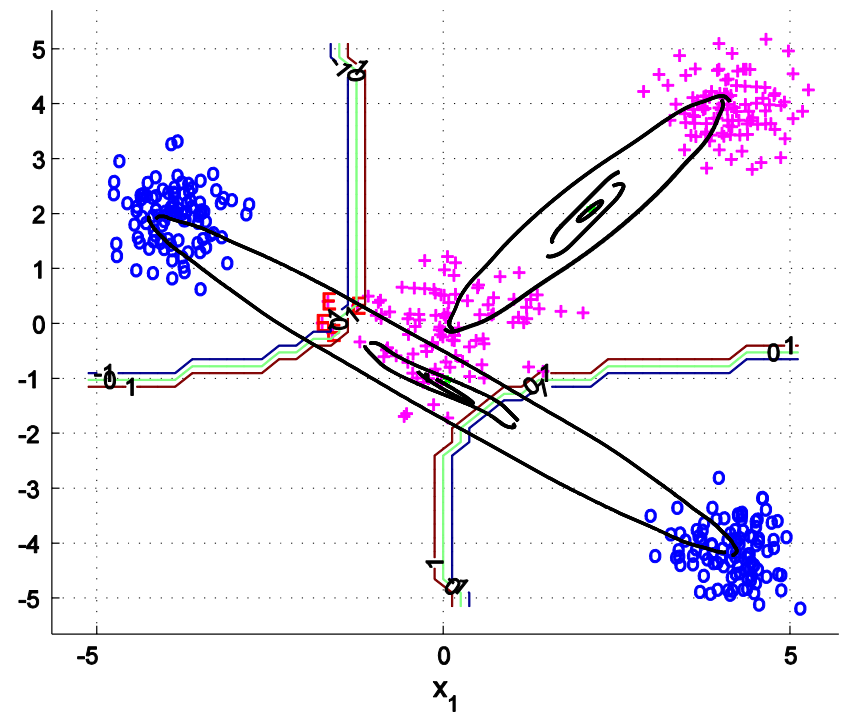
- Synthetic 1 dataset: two-dimensional; 100 train patterns (50 + 50); 400 test patterns (200 + 200)
- Using up to $M=10$ classifiers
- Test set error rate: AG ($M=10$) 6.5% ARBF ($M=3$) 1%

AG - Test set with 400 points. 10 weak classifiers with 26 errors



AG

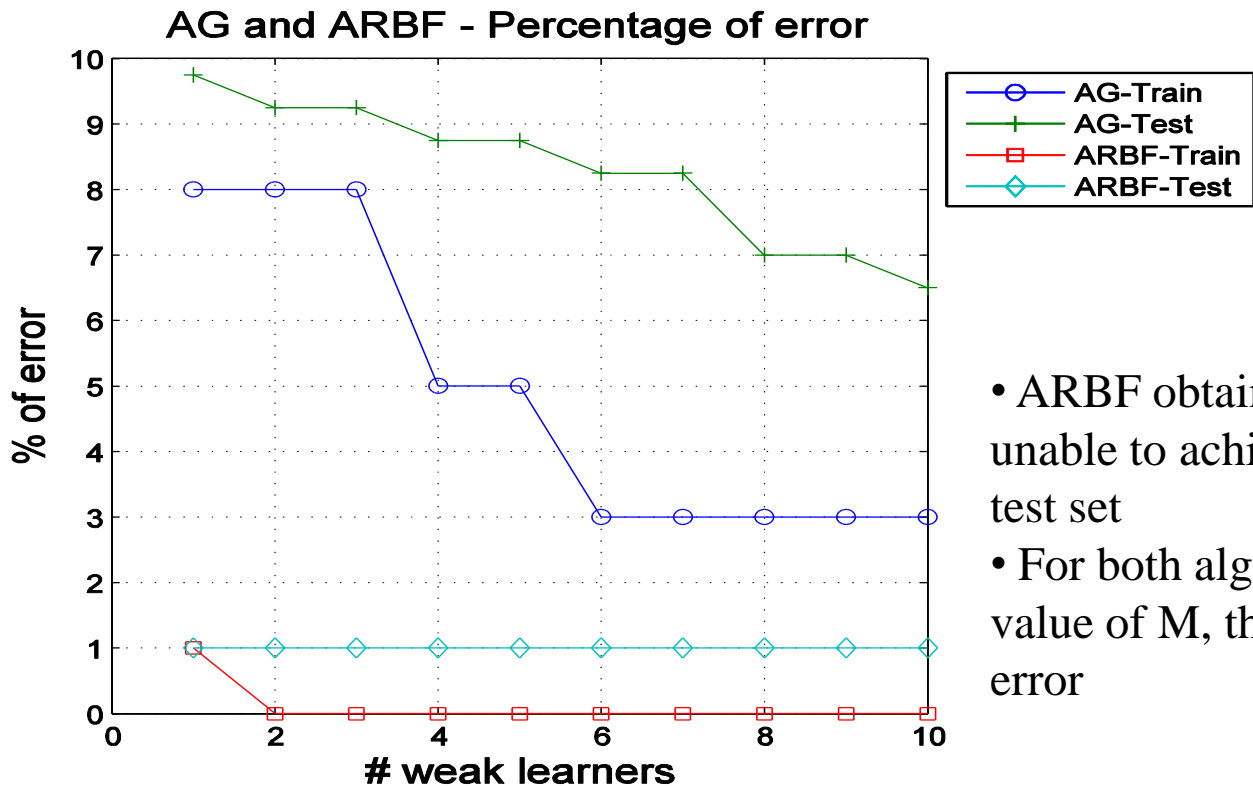
ARBF - Test set with 400 points. 3 weak classifiers with 4 errors



ARBF

4. Experimental results – synthetic 1

- Synthetic 1 dataset: two-dimensional; 100 train patterns (50 + 50); 400 test patterns (200 + 200)
- Using up to $M=10$ classifiers
- Training and test set error rate, as a function of M

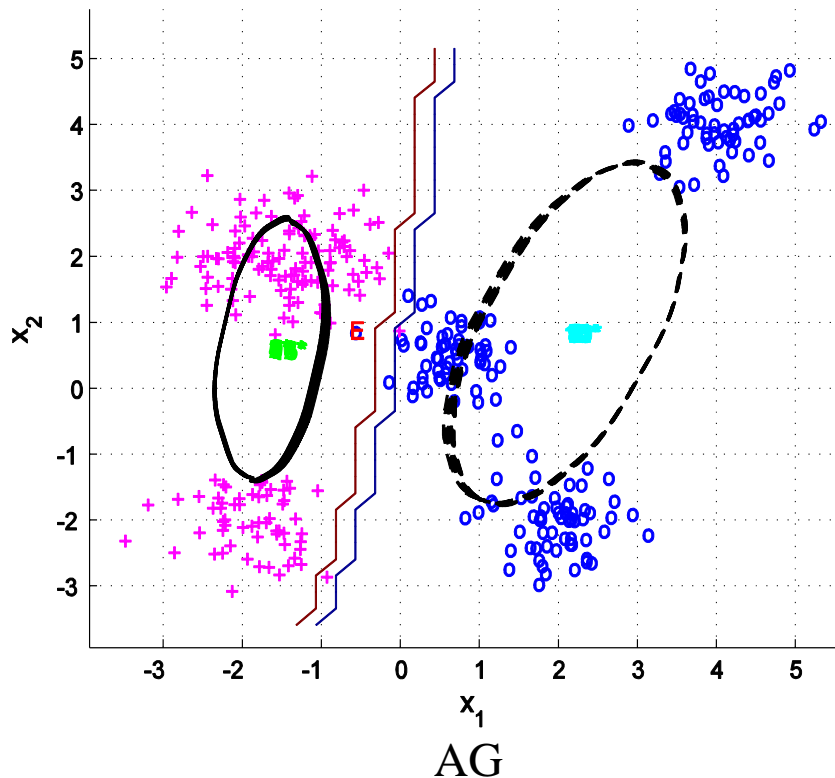


- ARBF obtains better results, but is unable to achieve zero errors on the test set
- For both algorithms, after a given value of M , there is no decrease on the error

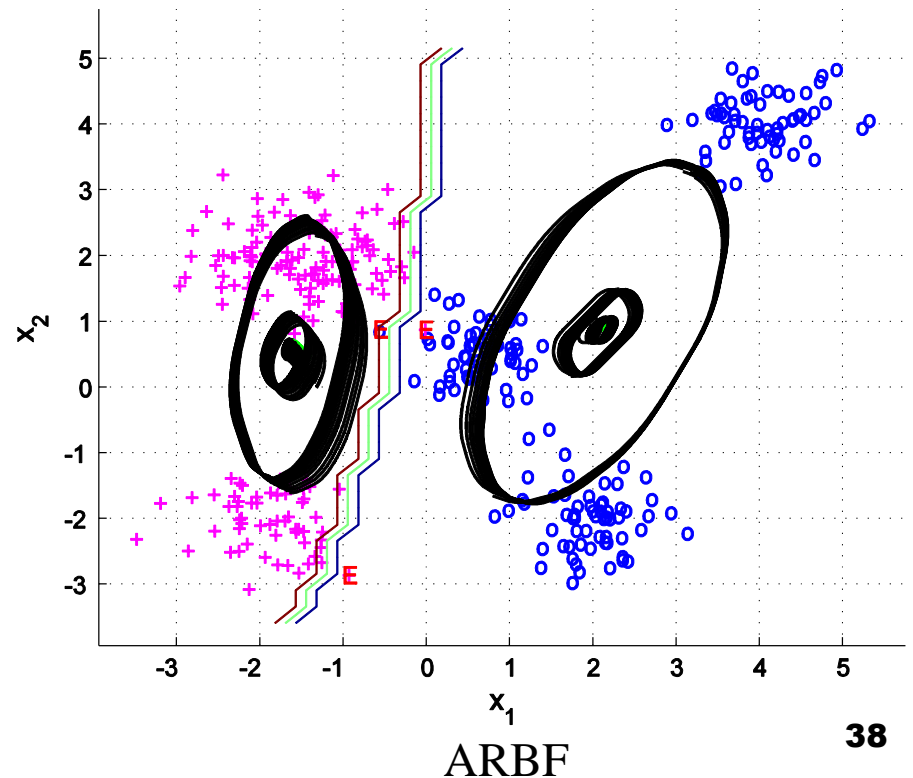
4. Experimental results – synthetic 2

- Synthetic 2 dataset: two-dimensional; 180 train patterns (90 + 90); 360 test patterns (180 + 180)
- Using up to $M=15$ classifiers
- Test set error rate: AG ($M=15$) 0.28% ARBF ($M=15$) 0.83%

AG - Test set with 360 points. 15 weak classifiers with 1 error

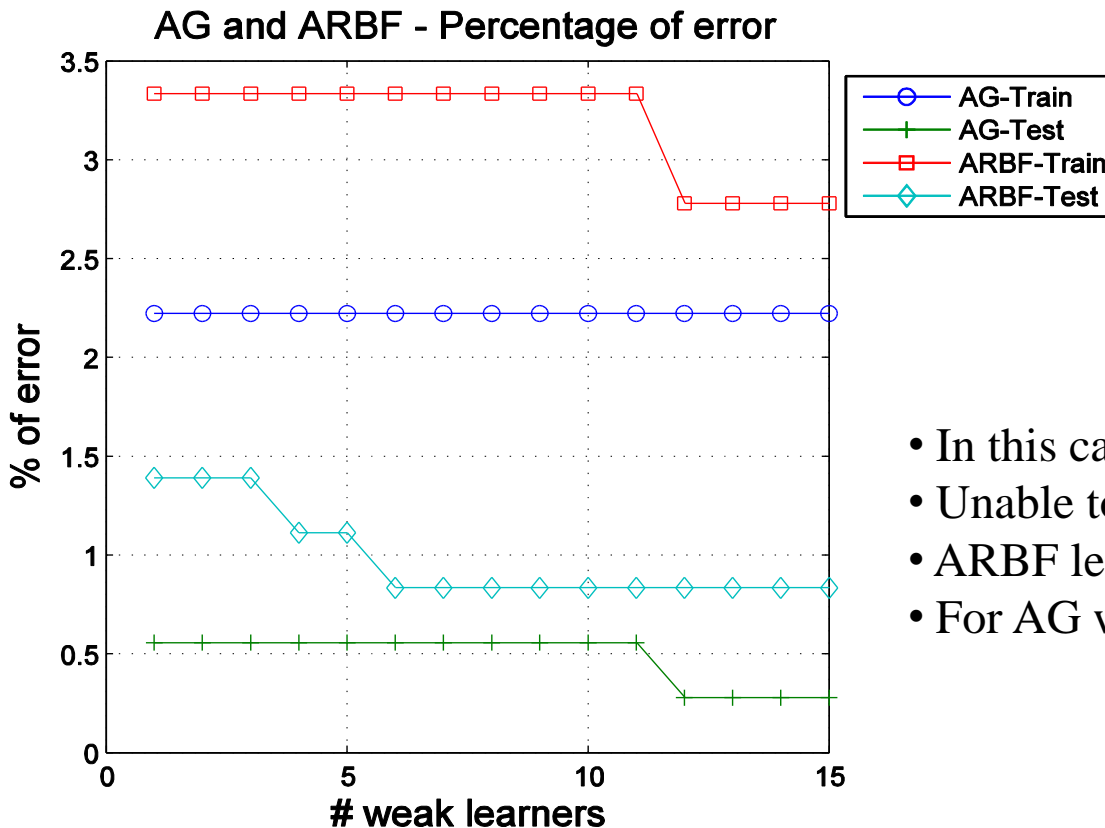


ARBF - Test set with 360 points. 15 weak classifiers with 3 errors



4. Experimental results – synthetic 2

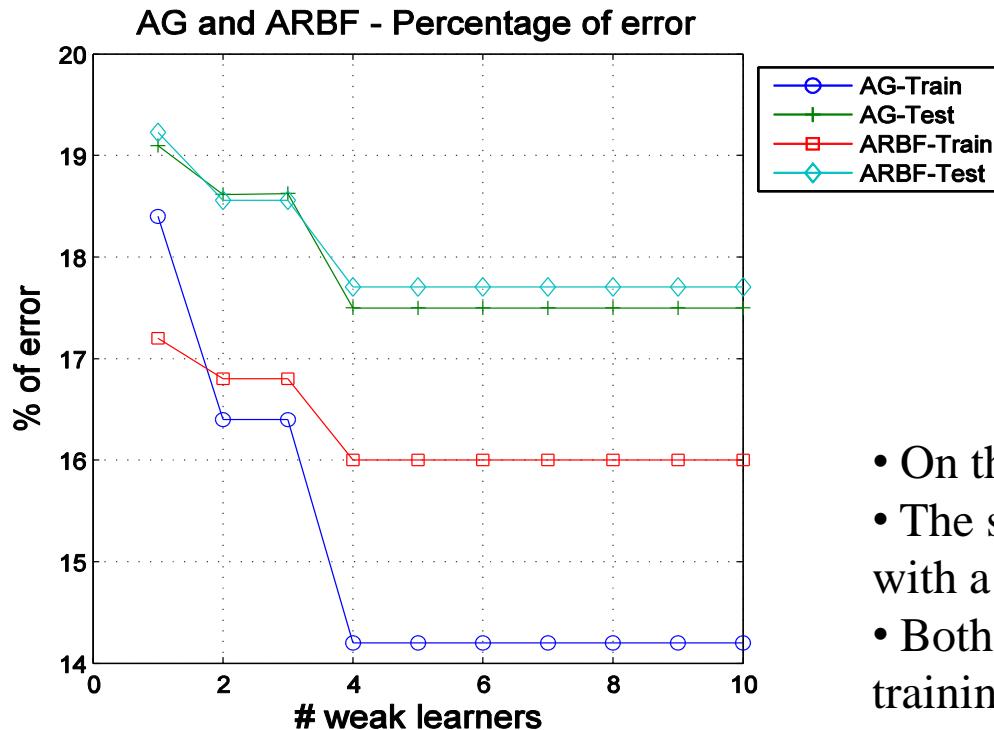
- Synthetic 2 dataset: two-dimensional; 180 train patterns (90 + 90); 360 test patterns (180 + 180)
- Using up to M=15 classifiers
- Training and test set error rate, as a function of M



- In this case, AG achieves better results
- Unable to achieve zero errors on the test set
- ARBF levels off with M=6
- For AG we have it with M=12

4. Experimental results – standard data sets

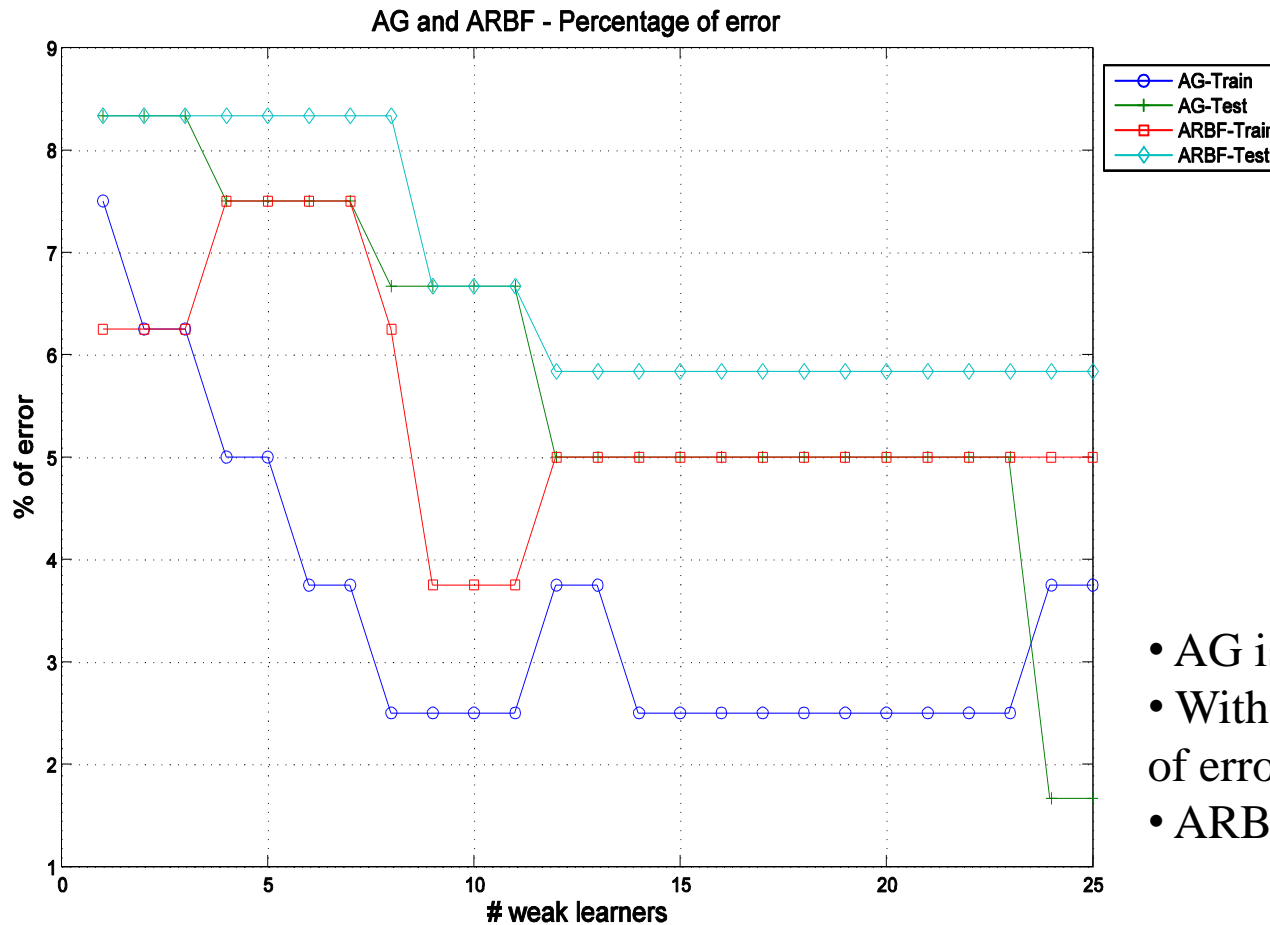
Dataset	Dimension	Train	Test
Kwok	2	200 ₍₁₎ + 300 ₍₀₎	4080 ₍₁₎ + 6120 ₍₀₎



- On the test set, AG is slightly better
- The same happens on the training set, but with a larger difference
- Both algorithms level off with $M=4$, on the training and test sets

4. Experimental results – standard data sets

Dataset	Dimension	Train	Test
Crabs	5	$40_{(1)}$ + $40_{(0)}$	$60_{(1)}$ + $60_{(0)}$



- AG is better than ARBF
- With $M=24$, AG attains 1.7% of error rate on the test set
- ARBF levels off with $M=12$

4. Experimental results – standard data sets

Dataset	Dim	AG Adaboost Generative		ATR Adaboost Trees Real		ATM Adaboost Trees Modest		SVM
		M	%	M	%	M	%	
Ripley	2	4	10.3	15	15.5	15	<u>10.1</u>	10.6
Kwok	2	4	17.5	80	13.6	80	<u>12.5</u>	11.7
Crabs	5	25	<u>1.7</u>	24	45.8	24	47.5	3.3
Phoneme	5	3	<u>20.9</u>	3	22.1	3	22.1	15.4
Pima	7	3	23.5	3	<u>23.2</u>	3	<u>23.2</u>	19.7
Abalone	8	3	<u>24.5</u>	3	24.7	3	24.7	20.9
Contraceptive	9	3	36.7	3	<u>34.4</u>	3	<u>34.4</u>	28.6
TicTacToe	9	4	<u>24.5</u>	4	28.2	3	33.2	1.7

M – the number of weak learners

Blue – the best test set error rate

Underline – the best boosting algorithm

4. Experimental results

- Application to face detection

Detector	AdaBoost Variant	Weak Learner
Viola-Jones (2001)	Discrete AdaBoost	Stub
Float Boost (2004)	Float Boost	1D Histograms
KLBoost (2003)	KLBoost	1D Histograms
Schneiderman	Real AdaBoost	One group of n-D Histograms

5. Concluding remarks

- Adaptive Boosting is a field of intensive research
- The main focus of research is for SSL
 - It seems to be “easy” to accommodate for unlabeled data
- Our tests, on SL, showed that:
 - Boosting of generative and RBF classifiers achieve performance close to boosting of trees, on standard datasets
 - Good performance with low training complexity, suited for embedded systems
 - AG and ARBF obtain results close to SVM, using a small number of weak learners
 - Typically, (the simpler) AG attains better results than ARBF
 - For high-dimensional datasets (with small training set) may be necessary to use diagonal covariances

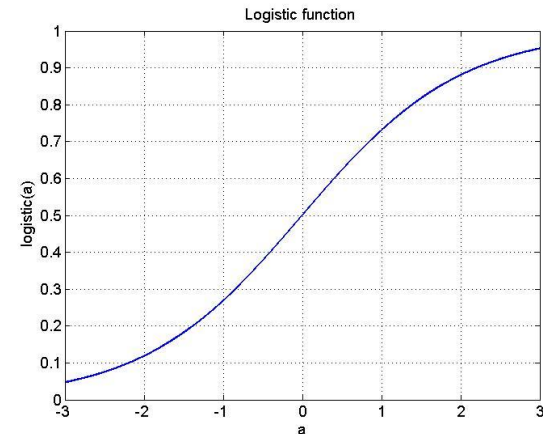
5. Concluding remarks: ongoing work

■ For SL:

- Establish (one more!) variant of AdaBoost, using the logistic function
- The logistic function gives us a degree of confidence on the classification

■ For SSL:

- Study and minor modifications of e algorithms
 - MixtBoost, SSMarginBoost, SemiBoost
- Extend our AdaBoost variant, proposed for SL, to SSL
 - using the logistic function properties to accommodate for unlabeled data



5. Concluding remarks: ongoing work

■ Logistic Function

$$l(a) = \frac{1}{1 + \exp(-a)}$$

