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

- \( H_1(x), \ldots, H_M(x) \) 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 $\alpha_m$

$$H(x) = \text{sign} \left( \sum_{m=1}^{M} \alpha_m H_m(x) \right)$$
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 $Z=\{z_1, z_2, \ldots, z_N\}$, 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

\[ Z = (z_1, z_2, ..., z_N) \]
2. Bootstrap: algorithm

**Input:**
- Training set \( Z=\{z_1, z_2, \ldots, z_N\} \), 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 \( n=\{1, \ldots, B\} \).

**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 $Z = \{ z_1, z_2, \ldots, z_N \}$, 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 $Z = \{z_1, z_2, \ldots, z_N\}$, 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 $n=\{1, \ldots, B\}$.

$$H(x) = \text{sgn} \left( \sum_{n=1}^{B} H_n(x) \right)$$
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 my a majority vote (3 classifiers)
2. Boosting: algorithm (classification)

Input:
• Training set $\mathbf{Z} = \{z_1, z_2, \ldots, z_N\}$, 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 $\mathbf{Z}$, obtaining $\mathbf{Z}^1$; train weak learner $H_1$ on $\mathbf{Z}^1$.

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

Step 3 - Select all samples from $\mathbf{Z}$ 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

\[ H(x) = \text{sgn}(y) \]

Final Classifier (step 4)

- \( L_1 \) samples (step 1)
- \( L_2 \) samples (step 2)
- \( L_3 \) samples (the ones on \( H_1 \) and \( H_2 \) differ) (step 3)

\( Z \)

\( Z^1 \)

\( Z^2 \)
3. AdaBoost Algorithm

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

- Consecutive classifiers are learned on weighted versions of the data.
- The weight is given by the (mis)classification of the previous classifiers.
- The next classifier focuses 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  
\[
H(x) = \text{sign}\left( \sum_{m=1}^{M} \alpha_m H_m(x) \right)
\]

1. Initialize the weights \(w_i\), with  
\[
w_i = \frac{1}{N} \quad i \in \{1, \ldots, N\}
\]
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 \(\text{err}_m\)
   c) Compute the contribution  
   \[
   \alpha_m = 0.5 \log \left( \frac{1 - \text{err}_m}{\text{err}_m} \right)
   \]
   d) Update the weights  
   \[
w_i \leftarrow w_i \exp\left( -\alpha_m I(y_i \neq H_m(x)) \right)
   \]
3. Output the ensemble  
\[
H(x) = \text{sign}\left( \sum_{m=1}^{M} \alpha_m H_m(x) \right)
\]
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

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

Indicator function:
- $-1$, different
- $1$, equal

Weight increases
Weight decreases
3. AdaBoost Algorithm

Comparison of Boosting (1989) and AdaBoost (1996)

<table>
<thead>
<tr>
<th>Feature</th>
<th>Boosting</th>
<th>AdaBoost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data processing</td>
<td>Random sampling without replacement</td>
<td>Weighting (no sampling)</td>
</tr>
<tr>
<td>Num. of classifiers</td>
<td>Three</td>
<td>Up to M</td>
</tr>
<tr>
<td>Decision</td>
<td>Majority vote</td>
<td>Weighted vote</td>
</tr>
</tbody>
</table>
3. AdaBoost Algorithm: variants

- Variants for SL

- Binary case

- Since 1999, several variants have been proposed

<table>
<thead>
<tr>
<th>Variants</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boosting (1989)</td>
<td></td>
</tr>
<tr>
<td>AdaBoost (1996)</td>
<td></td>
</tr>
<tr>
<td>Real AdaBoost (1999)</td>
<td></td>
</tr>
<tr>
<td>Margin Boost (2000)</td>
<td></td>
</tr>
<tr>
<td>Modest AdaBoost (2000)</td>
<td></td>
</tr>
<tr>
<td>AnyBoost (2000)</td>
<td></td>
</tr>
<tr>
<td>MarginBoost (2000)</td>
<td></td>
</tr>
<tr>
<td>KLBoost (2003)</td>
<td></td>
</tr>
<tr>
<td>FloatBoost (2004)</td>
<td></td>
</tr>
<tr>
<td>ActiveBoost (2004)</td>
<td></td>
</tr>
<tr>
<td>JensenShannon Boost (2005)</td>
<td></td>
</tr>
<tr>
<td>Infomax Boost (2005)</td>
<td></td>
</tr>
<tr>
<td>Emphasis Boost (2006)</td>
<td></td>
</tr>
<tr>
<td>Entropy Boost (2007)</td>
<td></td>
</tr>
<tr>
<td>Reweight Boost (2007)</td>
<td></td>
</tr>
<tr>
<td>LogitBoost (????)</td>
<td></td>
</tr>
<tr>
<td>Brown Boost (????)</td>
<td></td>
</tr>
<tr>
<td>Weight Boost (????)</td>
<td></td>
</tr>
</tbody>
</table>
3. AdaBoost Algorithm: variants

- Variants for SL

- Multiclass case

- Since 1997, several variants have been proposed

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AdaBoost.OC (1997)</td>
<td></td>
</tr>
<tr>
<td>AdaBoost.ECC (1999)</td>
<td></td>
</tr>
<tr>
<td>BoostMA (2005)</td>
<td></td>
</tr>
<tr>
<td>AdaBoost.ERP (2006)</td>
<td></td>
</tr>
</tbody>
</table>
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...!

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SemiBoost (2007)</td>
</tr>
</tbody>
</table>
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 $Z = \{z_1, z_2, \ldots, z_N\}$, 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 $w_i = 1/N$, $i = \{1, \ldots, N\}$

**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 $w_i \leftarrow w_i \exp \left( \sum_{m=1}^{M} y_i H_m(x) \right)$ 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 \( Z = \{z_1, z_2, \ldots, z_N\} \), 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 \( w_i = 1/N \), \( i = \{1, \ldots, N\} \)

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 \( w_i \leftarrow w_i \exp \left( y_i H_m(x) \right) \) and renormalize.

Step 3
Produce the final classifier.

\[
H(x) = \text{sgn} \left( \sum_{n=1}^{M} H_n(x) \right)
\]
AdaBoost variants

- Modest AdaBoost (MA) improves Gentle AdaBoost (GA) improves it, using an “inverted” distribution
  \[ \overline{w} = 1 - w \]
  - 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 \( w_i = 1/N, i=\{1,...,N\} \)

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 \( \overline{w} = 1 - w \) and renormalize.
  c) Compute
  \[
  P_m^+ = P_w(y = +1, H_m(x)) \\
  P_m^- = P_w(y = -1, H_m(x)) \\
  \overline{P}_m^+ = P_w(y = +1, H_m(x)) \\
  \overline{P}_m^- = P_w(y = -1, H_m(x))
  \]
  d) Set \( H_m(x) = P_m^+(1 - \overline{P}_m^+) - P_m^-(1 - \overline{P}_m^-) \)
  e) Update \( w_i \leftarrow w_i \exp \left( -y_i H_m(x) \right) \)

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
  \[ H_m(x) = P_m^{+1}(1 - \overline{P}_m^{+1}) - P_m^{-1}(1 - \overline{P}_m^{-1}) \]

- 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 $w_i = 1/N$, $i=1,...,N$

**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 $H_{r,m}(x)$ 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

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

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

$$
H(x) = \begin{cases} 
0, & p_0 G_0(x|\mu_0, \Sigma_0) > p_1 G_1(x|\mu_1, \Sigma_1) \\
1, & \text{otherwise}
\end{cases}
$$
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

\[ H(x) = \frac{1}{1 + \exp\left(-\beta_0 - \sum_{j=1}^{2} \beta_j G_j(||x - \mu_j||)\right)} \]
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)
    \[
    z_i^s = \frac{\hat{\alpha}_s \, G(x_i | \mu_s, \Sigma_s)}{\sum_{r=1}^{k} \hat{\alpha}_r \, G(x_i | \mu_r, \Sigma_r)}
    \]
    \[
    \hat{\alpha}_s = \frac{1}{N} \sum_{i=1}^{N} z_i^s
    \]
    \[
    \hat{\mu}_s = \frac{\sum_{i=1}^{N} x_i z_i^s}{\sum_{i=1}^{N} z_i^s}, \quad \hat{\Sigma}_s = \frac{\sum_{i=1}^{N} (x_i - \hat{\mu}_s)(x_i - \hat{\mu}_s)^T z_i^s}{\sum_{i=1}^{N} z_i^s}
    \]
  - Simultaneously, the output layer coefficients are learned by logistic regression (discriminative approach), by the Newton-Raphson step
    \[
    \beta \leftarrow \beta + (X^T W X)^{-1} X^T (y - p)
    \]

* A. Ferreira and M. Figueiredo, Hybrid generative/discriminative training of RBF Networks, ESANN2006
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%
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%
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

![Graph showing AG and ARBF - Percentage of error](image)

- 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

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Dimension</th>
<th>Train</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kwok</td>
<td>2</td>
<td>200 (1) + 300 (0)</td>
<td>4080 (1) + 6120 (0)</td>
</tr>
</tbody>
</table>

- 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

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Dimension</th>
<th>Train</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crabs</td>
<td>5</td>
<td>40&lt;sub&gt;(1)&lt;/sub&gt; + 40&lt;sub&gt;(0)&lt;/sub&gt;</td>
<td>60&lt;sub&gt;(1)&lt;/sub&gt; + 60&lt;sub&gt;(0)&lt;/sub&gt;</td>
</tr>
</tbody>
</table>

- 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

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Dim</th>
<th>AG Adaboost</th>
<th>ATR Adaboost Trees</th>
<th>ATM Adaboost Trees</th>
<th>SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Generative</td>
<td>Real</td>
<td>Modest</td>
<td></td>
</tr>
<tr>
<td>Ripley</td>
<td>2</td>
<td>4</td>
<td>10.3</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kwok</td>
<td>2</td>
<td>4</td>
<td>17.5</td>
<td>80</td>
<td>80</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Crabs</td>
<td>5</td>
<td>25</td>
<td>1.7</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Phoneme</td>
<td>5</td>
<td>3</td>
<td>20.9</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pima</td>
<td>7</td>
<td>3</td>
<td>23.5</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Abalone</td>
<td>8</td>
<td>3</td>
<td>24.5</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Contraceptive</td>
<td>9</td>
<td>3</td>
<td>36.7</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TicTacToe</td>
<td>9</td>
<td>4</td>
<td>24.5</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Detector</th>
<th>AdaBoost Variant</th>
<th>Weak Learner</th>
</tr>
</thead>
<tbody>
<tr>
<td>Float Boost (2004)</td>
<td>Float Boost</td>
<td>1D Histograms</td>
</tr>
<tr>
<td>KLBoost (2003)</td>
<td>KLBoost</td>
<td>1D Histograms</td>
</tr>
<tr>
<td>Schneiderman</td>
<td>Real AdaBoost</td>
<td>One group of n-D Histograms</td>
</tr>
</tbody>
</table>
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 existing 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

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