Combination Methods in Classifier Ensembles

Ludmila I. Kuncheva
School of Informatics
University of Wales, Bangor

mas00a@bangor.ac.uk
http://www.informatics.bangor.ac.uk/~kuncheva
Combination Methods in Classifier Ensembles

The plan

1. Introduction

2. Classifier selection

3. Classifier fusion for label outputs

4. Classifier fusion for continuous-valued outputs

5. Conclusions
<table>
<thead>
<tr>
<th><strong>Pattern Recognition</strong></th>
<th><strong>Machine Learning</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>classifier</td>
<td>learner</td>
</tr>
<tr>
<td>feature</td>
<td>attribute</td>
</tr>
<tr>
<td>object</td>
<td>example, instance</td>
</tr>
<tr>
<td>classifier output</td>
<td>hypothesis</td>
</tr>
<tr>
<td>class</td>
<td>class (reluctantly...)</td>
</tr>
</tbody>
</table>
Classifier ensembles - names

combination of multiple classifiers [Lam95, Woods97, Xu92, Kittler98]
classifier fusion [Cho95, Gader96, Grabisch92, Keller94, Bloch96]
mixture of experts [Jacobs91, Jacobs95, Jordan95, Nowlan91]
committees of neural networks [Bishop95, Drucker94]
consensus aggregation [Benediktsson92, Ng92, Benediktsson97]
voting pool of classifiers [Battiti94]
dynamic classifier selection [Woods97]
composite classifier systems [Dasarathy78]
classifier ensembles [Drucker94, Filippi94, Sharkey99]
bagging, boosting, arcing, wagging [Sharkey99]
modular systems [Sharkey99]
collective recognition [Rastrigin81, Barabash83] oldest
stacked generalization [Wolpert92]
divide-and-conquer classifiers [Chiang94]
pandemonium system of reflective agents [Smieja96] fanciest
change-glasses approach to classifier selection [KunchevaPRL93]
etc.
Classifier ensembles - philosophy

A classifier is any function $f \text{ (object description)} = \text{class label}$

- classifier
  - feature values
    - (object description)
  - class label
A classifier is any function $f(\text{object description}) = \text{class label}$
Classifier ensembles - philosophy

A classifier is any function $f \ (\text{object description}) = \text{class label}$

A neural network

feature values
(object description)

class label

combiner
Classifier ensembles - why ensembles then?...

a. because we like to complicate entities beyond necessity (anti-Occam’s razor)

b. because we are lazy and stupid and can’t be bothered to design and train one single sophisticated classifier

c. because democracy is so important to our society, it must be important to classification
Combination Methods in Classifier Ensembles

The plan

1. Introduction

2. Classifier selection

3. Classifier fusion for label outputs

4. Classifier fusion for continuous-valued outputs

5. Conclusions
Classifier ensembles - the two strategies
FUSION versus SELECTION

Combiner/Selector

- Feature values
- Class label

Classifier selection
Pick one
Use them all

Classifier fusion
Classifier ensembles - SELECTION

Why does it work?

$D_1$: All points are red

$D_2$: All points are blue

$D_3$

Note: All individual classifiers have accuracy close to 0.5

Note: Using $D_1$ in $R_3$, $D_3$ in $R_2$ and $D_2$ in $R_1$, the ensemble gives almost perfect accuracy
**Dynamic classifier selection algorithm (Woods et al.)**

Estimate the **local competence** and take the decision of the most competent expert. Local competence = $P(D \text{ is correct given the suggested label})$
Classifier ensembles - SELECTION
Mixture of Experts

\[ P(D_i \mid x), \quad i = 1, \ldots, L \]
Combination Methods in Classifier Ensembles

The plan

1. Introduction

2. Classifier selection

3. Classifier fusion for label outputs

4. Classifier fusion for continuous-valued outputs

5. Conclusions
An example: 9 single neural networks
An example: An ensemble -
The Average Combination Method

- Individual testing error 10.78%
- Ensemble testing error 9.50%
Suppose that all $L$ classifiers have the same accuracy $p > 0.5$.

<table>
<thead>
<tr>
<th>$p$</th>
<th>$L = 3$</th>
<th>$L = 5$</th>
<th>$L = 7$</th>
<th>$L = 9$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.6</td>
<td>0.648</td>
<td>0.683</td>
<td>0.710</td>
<td>0.733</td>
</tr>
<tr>
<td>0.7</td>
<td>0.784</td>
<td>0.837</td>
<td>0.874</td>
<td>0.901</td>
</tr>
<tr>
<td>0.8</td>
<td>0.896</td>
<td>0.942</td>
<td>0.967</td>
<td>0.980</td>
</tr>
<tr>
<td>0.9</td>
<td>0.972</td>
<td>0.991</td>
<td>0.997</td>
<td>0.999</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$p$</th>
<th>$L = 3$</th>
<th>$L = 5$</th>
<th>$L = 7$</th>
<th>$L = 9$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.6</td>
<td>0.900</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0.7</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0.8</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0.9</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
1. Majority (plurality) vote
2. Weighted majority vote
3. Naïve Bayes
4. Multinomial (BKS)
5. Probabilistic tree

**Weighted majority vote (the AdaBoost combination method)**

An example:
Label \( x \) using the weighted majority vote

Given: 5 classifiers with accuracies as in the table, labels for \( x: 1, 1, 3, 2, 2 \)

<table>
<thead>
<tr>
<th>( p )</th>
<th>0.78</th>
<th>0.72</th>
<th>0.80</th>
<th>0.68</th>
<th>0.74</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weight</td>
<td>1.27</td>
<td>0.94</td>
<td>1.39</td>
<td>0.75</td>
<td>1.05</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
\mu_1(x) &= 1.27 + 0.94 = 2.21 \\
\mu_3(x) &= 1.39 \\
\mu_2(x) &= 0.75 + 1.05 = 1.80
\end{align*}
\]

Assign label 1
T1. Assume that $p_i$ is the probability that classifier $D_i$ gives the correct label and also that $P(s | \omega_k) = \prod_{i=1}^{L} P(s_i | \omega_k), \forall k = 1, ..., c$

this being either $p_i$ or $(1 - p_i)$

Then the **weighted majority vote** with weights

$$w_i = \log \left( \frac{p_i}{1 - p_i} \right)$$

guarantees minimum classification error.
Naïve Bayes

Denote by $s = [s_1, ..., s_L], s_i \in \Omega$ the set of all class labels given by the $L$ classifiers.

**T2.** Assume that $p_i$ is the probability that classifier $D_i$ gives the correct label and also that

$$P(s | \omega_k) = \prod_{i=1}^{L} P(s_i | \omega_k), \forall k = 1, ..., c$$

this being either $p_i$ or $(1 - p_i)$

Then the Naïve Bayes combiner guarantees minimum classification error.

$P(s_i | \omega_k)$ are estimated from the confusion matrices
Multinomial (BKS) = look-up table

<table>
<thead>
<tr>
<th>s1</th>
<th>s2=1</th>
<th>s2=2</th>
<th>s2=3</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1=1</td>
<td>12/3/6</td>
<td>4/2/11</td>
<td>7/5/0</td>
</tr>
<tr>
<td>s1=2</td>
<td>2/3/1</td>
<td>4/15/2</td>
<td>3/9/5</td>
</tr>
<tr>
<td>s1=3</td>
<td>9/3/1</td>
<td>2/4/4</td>
<td>1/0/16</td>
</tr>
</tbody>
</table>

Number of training data points labelled as s1=1 and s2=1, which come from class 3.

$P(\omega_k | s), k = 1, \ldots, c$

Class labels given by classifier D1

Class labels given by classifier D2
Multinomial (BKS) = look-up table

Number of training data points labelled as $s_1=1$ and $s_2=1$, which come from class 3.

class labels given by classifier D1

$$P(\omega_k \mid s), k = 1, \ldots, c$$

class labels given by classifier D2

curse of dimensionality
Probabilistic tree (Bayesian network)

For any permutation of indices $1, \ldots, L$

\[
P(s \mid \omega_k) = P(s_{m_1} \mid \omega_k)P(s_{m_2} \mid s_{m_1}, \omega_k)P(s_{m_3} \mid s_{m_1}, s_{m_2}, \omega_k) \ldots P(s_{m_L} \mid s_{m_1}, \ldots, s_{m_{L-1}}, \omega_k)
\]

These disappear in Naive Bayes

Instead of assuming independence, we allow for first-order dependence (only)

\[
P(s \mid \omega_k) = P(s_{m_1} \mid \omega_k)P(s_{m_2} \mid s_{m_1}, \omega_k)P(s_{m_3} \mid s_{m_1}, s_{m_2}, \omega_k) \ldots P(s_{m_L} \mid s_{m_1}, \ldots, s_{m_{L-1}}, \omega_k)
\]

The problem is to find the permutation of indices $m_1, m_2, \ldots, m_L$ and the corresponding pair indices $m_{j(1)}, m_{j(2)}, \ldots, m_{j(L)}$
**T3.** Assume that

\[ P(s | \omega_k) = \prod_{i=1}^{L} P(s_i | s_{j(i)}, \omega_k), \forall k = 1, ..., c, j(i) \in \{1, ..., i-1\} \]

Then the **probabilistic tree** combiner guarantees minimum classification error.

The permutation of indices is derived by a minimum spanning tree algorithm.

\[ P(s_i | s_{j(i)}, \omega_k) \] are estimated from data.
Number of parameters needed to operate

- **Multinomial (BKS)**: $c^L$
- **Probabilistic tree**: $c \cdot [(L-1)c^2 + L]$
- **Naive Bayes**: $Lc^2$
- **Weighted majority vote**: $L$
- **Majority (plurality) vote**: none
Combination Methods in Classifier Ensembles

The plan

1. Introduction

2. Classifier selection

3. Classifier fusion for label outputs

4. Classifier fusion for continuous-valued outputs

5. Conclusions
The output of the $L$ classifiers can be organised in a “decision profile”, e.g.,

\[
\begin{array}{cccc}
\omega_1 & \omega_2 & \omega_3 & \omega_4 \\
D_1 & 0.1 & 0.4 & 0.3 & 0.2 \\
D_2 & 0.3 & 0.3 & 0.3 & 0.1 \\
D_3 & 0.2 & 0.1 & 0.7 & 0.0 \\
\end{array}
\]

$\hat{P}_3(\omega_2 \mid \mathbf{x})$
Simple combiners: Where do they come from?

\[ \mu_k(x) = F(d_{1,k}(x), d_{2,k}(x), \ldots, d_{L,k}(x)) \]

Intuition!

<table>
<thead>
<tr>
<th>Class</th>
<th>(D_1)</th>
<th>(D_2)</th>
<th>(D_3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(D_1)</td>
<td>0.1</td>
<td>0.4</td>
<td>0.3</td>
</tr>
<tr>
<td>(D_2)</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
</tr>
<tr>
<td>(D_3)</td>
<td>0.2</td>
<td>0.1</td>
<td>0.7</td>
</tr>
</tbody>
</table>

Class \(\omega_2\) is supported at least that much by everybody.

There is at least one expert who thinks that class \(\omega_2\) should be supported that much.

Find a “consensus” degree of support for class \(\omega_2\).

Intuition!
Simple combiners: Where do they come from?

\[ \mu_k(x) = F(d_{1,k}(x), d_{2,k}(x), \ldots, d_{L,k}(x)) \]

**Assumptions and simplifications**

Conditionally independent representations

\[ x = [x_1 \ x_2 \ x_3 \ x_4 \ x_5 \ x_6 \ x_7 \ x_8]^T \]

Assume

\[ p(x | \omega_k) = \prod_{i=1}^{L} p(x^{(i)} | \omega_k) \]

leads to the

**Product rule**
Simple combiners: Where do they come from?

$$\mu_k(x) = F(d_{1,k}(x), d_{2,k}(x), \ldots, d_{L,k}(x))$$

Assumptions and simplifications

Conditionally independent representations

Assume

$$p(x | \omega_k) = \prod_{i=1}^{L} p(x^{(i)} | \omega_k)$$

Curiously, this same assumption leads to the

Sum (mean) rule

leads to the

Product rule
Simple combiners: Where do they come from?

\[ \mu_k(x) = F(d_{1,k}(x), d_{2,k}(x), \ldots, d_{L,k}(x)) \]

**A Bayesian perspective**

Depending on how we perceive the probabilities in the decision profile, we can derive

- the product rule
  \[ w_i = p_i \]

- the sum (mean) rule
  \[ w_i = p_i (1 - p_i)^{(1-p_i)} \]

- the weighted sum (mean) rule
  \[ w_i = \frac{1}{(1 - p_i)} \]
Fuzzy integral as a combiner

Take the $\alpha$-cuts

$\alpha = 0.0; \{D_1, D_2, D_3\}$
$\alpha = 0.1; \{D_1, D_2, D_3\}$
$\alpha = 0.2; \{D_2, D_3\}$
$\alpha = 0.3; \{D_2\}$
$\alpha = 1.0; \{\emptyset\}$

% Importance of the combination of votes

Choose $\omega_3$
Decision templates

Expand the decision profile $DP(x)$

$y = [d_{1,1}(x), d_{1,2}(x), ..., d_{L,c}(x)]^T$, $y \in \mathbb{R}^{L \times c}$

call this the intermediate feature space.

Decision templates =

Nearest Mean Classifier in the intermediate feature space
Any classifier can be used here

Intermediate feature space

$DP(x)$

Classifiers

1. Simple combiners
2. Fuzzy integral
3. Decision templates
4. Weighted average
1. Simple combiners
2. Fuzzy integral
3. Decision templates
4. Weighted average

Weighted average

\[ \mu_k(x) = \sum_{i=1}^{L} w_i d_{i,k}(x) \]

\[ \mu_k(x) = \sum_{i=1}^{L} w_{i,k} d_{i,k}(x) \]

\[ \mu_k(x) = \sum_{i=1}^{L} \sum_{j=1}^{c} w_{i,k,j} d_{i,j}(x) \]

Uses all of the features in the intermediate feature space


Combination Methods in Classifier Ensembles

The plan

1. Introduction

2. Classifier selection

3. Classifier fusion for label outputs

4. Classifier fusion for continuous-valued outputs

5. Conclusions
Which is the best combiner?

As usual: Different combiners are “best” for different problems

Wrong question!

The right question:
How do we choose the best combiner for our data?
And another good question: To Train or not to Train?

<table>
<thead>
<tr>
<th>Option</th>
<th>Combiner Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train the classifiers on Z</td>
<td>minimum, maximum, sum, product, trimmed mean, majority vote</td>
</tr>
<tr>
<td>Do not train the combiner</td>
<td></td>
</tr>
<tr>
<td>Train the classifiers on Z</td>
<td>weighted average, weighted product, weighted majority vote, BKS, naïve Bayes, DTs, fuzzy integral</td>
</tr>
<tr>
<td>Train the combiner on Z</td>
<td></td>
</tr>
<tr>
<td>Train the classifiers on Z, Train the combiner on Z</td>
<td>mixture-of-experts, stacked generalization</td>
</tr>
<tr>
<td>Train the classifiers on Z, Train the combiner on V</td>
<td></td>
</tr>
<tr>
<td>Train the classifiers and the combiner on Z - together</td>
<td></td>
</tr>
</tbody>
</table>
And one final question: To fuse or to select?

most combination methods are in the middle anyway

... or

switch between selection and fusion depending on the confidence in estimating the competences.
Conclusions - Where Next?

Any classifier can be trained in the intermediate feature space to serve as the combiner. Do we stop there or build further layers?

There are many possibilities to build the ensemble. How do we choose a combiner to make the most of the design of the individual members?

Catch the Americans...
And this is what my grateful students are planning for me...