**Combined Classifiers**

**Several Classifiers in Same Feature Space**

Stacked Combining
Same Feature Space

\[ S_3(x) = 0 \]
\[ S_2(x) = 0 \]
\[ S_1(x) = 0 \]

**How to combine?**

**Several Classifiers in Different Feature Spaces: Parallel Combining**

**Parallel Combining Different Feature Spaces**

**How to combine?**

**Multiple Classifier Sources**

Different feature spaces: Face, Voice, Fingerprint

Different training sets: Sampling, Bootstrapping

Different classifiers: \( k_{-}NN \), Bayes Normal, Dec. Tree, SVC, Neural Net

Different architectures: Neural Net: \#Layers, \#Units, Transfer function.

Different parameter values: \( k \) in \( k_{-}NN \), kernel in SVC, pruning in Dec. Tree

Different initializations: Neural Net

**Combining Classifiers Architecture**

Base classifiers
Same / different models
Same / different training sets

Classifier 1

Classifier 2

Classifier 3

Classifier 4

Combining Classifier

Fixed / trained rules

Revision

Compare Neural Network!
Strategic Reasons for Multiple Classifiers

Multiple Sensors: Multiple Feature Spaces
No Unique Data Model: Multiple Feature Spaces Multiple Classifiers
Multi-modal data: Multiple Partial Solutions
Unstable Classification: Multiple Bootstrapped Training Sets
Classifier Economy: Multiple Initializations

How do we know how sure they are?
How do we know their expertise?

Two Opposite Multiple Classifier Strategies

How to generate base classifiers, given how to combine them (e.g. bagging, random subspace method)

Given base classifiers; how to combine them?

Experts

Voting

Number of votes for class A
\[
\frac{1}{n} \sum_{i} \text{Prob}(A|x)
\]

Majority rule:
Base classifier distribution \(\leftrightarrow\) posterior probability
### Posterior Probabilities

\[
\text{Prob} (x \in A | x) = \frac{p_A f_A(x)}{p(x)}
\]

### Posterior Probabilities for Arbitrary Classifiers: Normalization

\[
p(x) = \text{Prob} (x \in A | y = S(x)) = \frac{p_A f_A(y)}{f(y)}
\]

### Combining Different Representations - Different Areas of Expertise

- **Color**
- **Shape**

Base classifier \( j \) posterior probabilities for class A: \( y_{Aj} = \text{Prob}_j(A|x_j) \)

**Product Rule:** \( y_A = \prod \text{Prob}_j(A|x_j) \), \( y_B = \prod \text{Prob}_j(B|x_j) \)

Useful for 'independent' feature spaces (logical 'AND', experts should agree)

**Minimum Rule:** \( y_A = \text{Min}\{ \text{Prob}_j(A|x_j) \}, \ y_B = \text{Min}\{ \text{Prob}_j(B|x_j) \} \)

Assign according to ‘least objecting expert’

### Combining Different Estimates - Differently Trained Experts

**Sum (Mean) Rule:** \( y_A = \sum \text{Prob}_j(A|x), \ y_B = \sum \text{Prob}_j(B|x) \)

Useful for improved estimates of posterior probabilities

Also: **Median and Majority Voting**

Improvement by averaging out mistakes of experts
**Fixed combining rules**

- **Product, Minimum**
  - Independent feature spaces
  - Different areas of expertise
  - Error free posterior probability estimates

- **Sum (Mean), Median, Majority Vote**
  - Equal posterior-estimation distributions in same feature space
  - Differently trained classifiers, but drawn from the same distribution
  - Bad if some classifiers (experts) are very good or very bad

- **Maximum**
  - Trust the most confident classifier / expert
  - Bad if some classifiers (experts) are badly trained

---

**Trained Combining Classifier**

- **Base classifiers**
  - $y_{Aj} = \text{Prob}(A|x)$, $y_{Bj} = \text{Prob}(B|x)$

- **Classifier 1**
- **Classifier 2**
- **Classifier 3**
- **Classifier 4**

- **Combining Classifier**
  - Treat outputs as features: $y = (y_{A1}, y_{A2}, y_{A3}, y_{A4}, y_{B1}, y_{B2}, y_{B3}, y_{B4})$

- **Trained combiner**
  - Training set: $\{y_1, y_2, \ldots, y_m\}$

- **General rules** neglect the classification-confidence characteristic of the base classifier outputs, as they are treated as general feature values.

---

**Special Characteristics of the Combining Classifier Input Space**

- **Graph**:
  - 1. Features are ‘directed’
  - 2. Class distributions far for normal

- Decision Templates:
  - Fisher
  - Bayes Normal
  - Decision Tree

---

*Kuncheva, PR-23(2), 2001*
Combining 10 Bootstrapped Nearest Mean Classifiers

Example

Biased Outputs Base Classifiers

True error (= output evaluation set base classifiers)

Classification Error

Bias

Apparent error (= output training set base classifiers)

Should not be used for training combiner

Training set size

Combining Differently Trained Classifiers

True error (= output evaluation set base classifiers)

Classification Error

Undertrained (no evaluation set needed)

Well trained

Overtrained

(use evaluation set for training combiner)

Apparent error (= output training set base classifiers)

Training set size

See also S. Raudys, MCS2002
Independent Training Set Combining Classifier

Base classifiers

 Classifier 1
 Classifier 2
 Classifier...
 Classifier n

Train classifiers on a

Training set base classifiers

Trained combiner

Combining Classifier

b*(a*[c1 c2 c3 c4]*classc*fisherc)

b*(a*[c1 c2 c3 c4]*classc*fisherc)

Stacked Combining
Same Feature Space

Note Different Dimensionalities

Design set

Training base classifiers
Dimensionality = # features

Training output combiner
Dimension = #classes x #base classifiers

An equal split may not be the best!

Possible Strategies:

1 Use just a single training set.
   Train the base classifiers carefully, avoiding overtraining.
   Fixed combining rules may work.

2 Use just a single training set.
   Train the base classifiers weakly.
   The same training set may be used for the combining classifier.

3 Separate the available training sets into two parts (may be improved by 4).
   Use one part for training the base classifiers.
   Some overtraining is not a problem.
   Use the other part for training the combining classifier.

4 Use stacked generalisation.

Parallel versus Sequential Classifier Combining

input features

Classifier 1
Classifier 2
Classifier 3

Decision Tree

Equivalent with:

more features, more complex classifiers

objects to be classified

classifier 1

easy objects

classifier 2

more difficult objects

classifier 3

most difficult objects
**Special Training Rules**

- **Global Selection**: Select the best base classifier / fixed combining rule
- **Calibration**: Scale base classifier outputs in a similar way
- **Local Selection**: Select the best base classifier for the object at hand
- **Decision Templates**: Similar to the more general Nearest Mean Classifier

Other possibilities that use the specific character of the base classifier outputs for training?

**Conclusions on Fixed versus Trained Rules for Combiners**

Fixed rules are hardly ever theoretically optimal, but perform sometimes surprisingly good.
Trained rules can be optimal for large training sets.
Use of the same training set might be good for well / undertrained base classifiers.
Different training sets are needed for well / overtrained base classifiers.
How to split the total design set over training sets needs more study.
‘Decision templates’ is a good training rule, unless we have many base classifiers.
Special purpose combiners are to be developed.

---

**Bagging and Boosting**

- **Training Set**
  - Sample_1 → Classifier 1
  - Sample_2 → Classifier 2
  - Sample_n → Classifier n

**Base classifiers**

Generate many (e.g. 200) classifiers in the same space by resampling and retraining

**Fixed/Trained Combiner**

Bagging: use bootstrapping
Boosting: emphasize difficult objects

**Example: Banana (2 Features, 100 Objects)**

Banana Set

- **Feature 1**
- **Feature 2**

-10 -5 0 5
-8 -6 -4 -2 0 2 4
-6 -4 -2 0 2 4
-10 -5 0 5
A Set of 10 Linear Classifiers

Combining by Majority Voting

Maximum Combiner

Sum Combiner
Fisher Combiner

Discussion

- Combining simple base classifiers → complicated decision boundary
- Base classifiers should be different
- Trained combiners seem better, but may overtrain

Bagging and Boosting Strategy

**Bagging**
1. Subsample the dataset

**Boosting**

- Weighting difficult objects

2. Train simple base classifiers

<table>
<thead>
<tr>
<th>Decision Trees</th>
<th>Linear Classifiers</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Weak Classifier</strong> (Fisher)</td>
<td></td>
</tr>
</tbody>
</table>

3. Combine

| Voting, Sum | Weighted Voting (Fisher) |

Bagging Algorithm

1. **Bootstrap** the data set
2. **Train** the base classifier
3. **Goto 1** as long as needed
4. **Combine** all base classifiers
**Adaboost Algorithm**

1. **Sample** the training set according to a set of object weights (initially equal)
2. Use it for **training a** simple (weak) **classifier** \( w_i \)
3. **Classify** the entire data set, using the weights, **error** \( \varepsilon_i \)
   - Store classifier weight \( \alpha_i = 0.5 \log((1-\varepsilon_i)/\varepsilon_i) \)
4. **Multiply weights** of erroneously classified objects with \( \exp(\alpha_i) \)
   - Multiply weights of correctly classified objects with \( \exp(-\alpha_i) \)
5. **Goto 1** as long as needed
6. **Final classifier**: weighted voting, weights \( \alpha_i \)

*Schapire, Freund et al. 1999/99*

---

**Discussion on Base Classifiers**

Are to be combined,

Simple, not overtrained, especially not for trained combiners

Many: fast training, fast execution

Soft outputs might be helpful

Traditional: decision trees, decision stumps, linear, quadratic

Weak classifiers: simple, should do something,
   - not sufficient for the problem,
   - large bias, large variance

---

**Boosting: Emphasize Difficult Objects**

Dataset  \[ \text{Step 1} \]  \[ \text{Step 2} \]  \[ \text{Step 3} \]

\[ \text{Step 4} \]  \[ \text{Step 5} \]  \[ \text{Step 6} \]  \[ \text{Step 7} \]

\[ \text{Step 8} \]  \[ \text{Step 9} \]  \[ \text{Step 10} \]  \[ \text{Final} \]

---

**Adaboost - 2D Example**

100 dec. stumps, \( w \)vote

10 dec. stumps, Fisher

10 Fisher, Fisher

\[ w = \text{adaboostc}(a, \text{stumpc}, 100) \]

\[ w = \text{adaboostc}(a, \text{fisherc}, 10, \text{fisherc}) \]

\[ w = \text{adaboostc}(a, \text{stumpc}, 10, \text{fisherc}) \]
Decision stumps combined by weighted voting works!

For small # boosting steps, trained Fisher combiner improves

The Fisher base classifier improves decision stumps
Adaboost Example: Sonar (60 Features, 208 Objects)

For small # boosting steps, trained Fisher combiner improves

Adaboost Example: Heart (13 Features, 297 Objects)

Single classifiers are not improved by boosting

Boosting Observations

Resampling strategy:
The boosting principle may work for more difficult datasets

Base classifiers:
The use of weak base classifiers may be improved by stronger classifiers

Combiner:
Weighted voting performs well.
The trained Fisher combiner does better for small sets of base classifiers than the weighted voting.

Bagging and Boosting, Conclusions and Questions

Good for more complicated datasets with an unknown model (type of decision boundary).

How many base classifiers are needed?
What base classifiers may construct what final classifier?
Trained combiners like Fisher sometimes to be Preferred?