5. Building Classifier Ensembles

4.1. Error correcting output code (ECOC) ensembles


4.3. Bagging, AdaBoost and Random Forests

4.4. Selection + Fusion: Random Oracle
The text content of the image is as follows:

### Stages in Building a Classifier Ensemble

1. Optimise the combiner

2. Use error-correcting output codes (ECOC)

3. Try different classifier models e.g., decision trees, neural networks, nearest neighbour, etc.

4. Train each classifier on a different subset of features

5. Alter the training data: sample from the whole data set, inject noise (bagging, boosting, random forests)

---

### Error Correcting Output Codes (ECOC)

\( \Omega = \{ \omega_1, \omega_2, \omega_3, \omega_4, \omega_5, \omega_6 \} \)

multi-class problem (a polychotomy)

\( \Omega = \{ (\omega_1 \cup \omega_2 \cup \omega_3), (\omega_2 \cup \omega_3 \cup \omega_6) \} \)

2-class problem (a dichotomy)

Transform the multi-class problem into 2-class problems (parallel dichotomizers).

The standard approach:

- **one-per-class**
  \( \omega_h \)
  against
  \( \omega_1 \cup \ldots \cup \omega_{h-1} \cup \omega_{h+1} \ldots \cup \omega_c \)

- **error-correcting output codes (ECOC)**

- Different classifier models/training

- Different subsets of features

- Different training data
Error Correcting Output Codes (ECOC)

Number or rows = \( c \), i.e., one code word per class.

Number of columns (bits) = \( L \), i.e., Each “bit” is supplied by a classifier.

\[
\begin{array}{cccc}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0
\end{array}
\]

code word for class \( \omega_1 \)

code word for class \( \omega_2 \)

Hamming distance = 2

d - minimum Hamming distance between any two code words.

Number of possible corrections: \( \left\lceil \frac{d - 1}{2} \right\rceil \)

For one-per-class, \( \left\lceil \frac{2 - 1}{2} \right\rceil = 0 \), no error correcting power

How to design a good ECOC?

1. As large as possible Hamming distance between the rows (code words for the classes)

2. Low correlation between the columns, i.e., the classifiers should have as different as possible tasks to solve in order to create diversity in the ensemble.

Note: Impossible for small number of classes. Example \( c = 3 \)

<table>
<thead>
<tr>
<th>Class</th>
<th>( f_1 )</th>
<th>( f_2 )</th>
<th>( f_3 )</th>
<th>( f_4 )</th>
<th>( f_5 )</th>
<th>( f_6 )</th>
<th>( f_7 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \omega_1 )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>( \omega_2 )</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>( \omega_3 )</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Possible number of different code words for $c$ classes \[ 2^{c-1} - 1 \]

Methods for constructing ECOC (Dietterich and Bakiri)

1. Exhaustive codes
2. Column selection from exhaustive codes \(8 \leq c \leq 11\), Formulate and solve an optimization problem.
3. Random assignment (Schapire) Pick each bit in each codeword with probability 50/50.
4. Random assignment - even split (Schapire) Pick at random code words for which the number of 0s is equal to the number of 1s.
5. Random hill climbing (Dietterich and Bakiri) \(c > 11\)
6. Diversity-based - Use “diversity” and pseudo-random search to build the ECOC (Kuncheva)
7. Based on a hierarchical structure of the classes (Pujol et al 2006)

---

**Stages in Building a Classifier Ensemble**

1. Optimise the combiner
2. Use error-correcting output codes (ECOC)
3. Try different classifier models e.g., decision trees, neural networks, nearest neighbour, etc.
4. Train each classifier on a different subset of features
5. Alter the training data: sample from the whole data set, inject noise (bagging, boosting, random forests)
Classifier fusion and selection

The output label

Combiner/Selector

Pick one

Classifier selection

Use them all

Classifier fusion

Classifier Classifier Classifier ...

The input vector

The combiner

Error-correcting output codes (ECOC)

Different classifier models/training

Different subsets of features

Different training data

Dynamic classifier selection

Estimate the local competence and take the decision of the most competent expert.

D_1
Neural network

D_2
k-nn

D_3
Naive Bayes

The combiner

Error-correcting output codes (ECOC)

Different classifier models/training

Different subsets of features

Different training data
The classifier ensembles (C. Kuncheva, School of CS, Bangor University, UK).

Stages in Building a Classifier Ensemble

1. Optimise the combiner

2. Use error-correcting output codes (ECOC)

3. Try different classifier models
e.g., decision trees, neural networks, nearest neighbour, etc.

4. Train each classifier on a different subset of features --- later...

5. Alter the training data: sample from the whole data set, inject noise (bagging, boosting, random forests)
Bagging

For an ensemble of $L$ classifiers, take $L$ bootstrap samples from $Z$ and train a classifier on each sample.

AdaBoost

For an ensemble of $L$ classifiers, take $L$ bootstrap samples from $Z$ and train a classifier on each sample. In each consecutive sample, give higher priority to points that have been mislabelled by the classifiers built hitherto.

AdaBoost is great but does not cope well with noise.
For an ensemble of $L$ classifiers, take $L$ bootstrap samples from $Z$ and train a random tree classifier on each sample. To split a node of the tree, select a the best feature from a random subsample of size $M$ from the original feature set (e.g., $M = 5$).

... construct the rest of the tree in the same way

... construct the bagging ensemble using such trees

---

Random forest (Breiman)

1. Optimize the combiner
2. Use error-correcting output codes (ECOC)
3. Try different classifier models e.g., decision trees, neural networks, nearest neighbour, etc.
4. Train each classifier on a different subset of features
5. Alter the training data: sample from the whole data set, inject noise (bagging, boosting, random forests)
Selection (use one)

Fusion (use all)

- mixture of experts
- subset selection
- weighted majority vote
- weighted sum

Selection + Fusion: Random Oracle

How about?...

Selection + Fusion

Selection (use one)
Kuncheva L.I. and J.J. Rodriguez, Classifier ensembles with a random linear oracle, *IEEE Transactions on Knowledge and Data Engineering, 19* (4), 2007, 500-508

This is what happens for 1 ensemble member:

The split is randomly chosen and then FIXED.

... construct the rest of the ensemble in the same way.
Selection + Fusion: Random Oracle

Examples of ensemble members

Quick to train and run

Increases diversity and ensemble accuracy for many ensemble methods

Does not compromise the individual accuracy

Can be used with any classifier (sub-classifier)

Can be combined with other ensemble methods, e.g., bagging

The split of the space can be done in different ways, e.g., “Spherical Oracle” (Rodriguez and Kuncheva 2007)
You can't beat the students...

The Joke

Or maybe you can...?

The Joke
**Selection + Fusion: Random Oracle**

**PROBLEM:** One possible explanation for the success of the random oracle is that the sub-classifiers are solving simpler classification problems than a single classifier would, because they only work in half of the feature space.

One measure of the complexity of the classification problem is based on Minimum Spanning Tree (MST). Construct the MST on the whole data set and count the number of branches that connect points with different class labels. The larger this number, the more complex the data.

Consider a random linear oracle applied on a data set Z, such that there is at least one point on each side of the line. Here is the hypothesis:

**Hypothesis:** The MST data complexity on each side of the line does not exceed the data complexity of the original set Z.

*Can you prove or disprove it?*

---

**Puzzle**

**CHALLENGE:** Matlab (nothing to do with the course 😊)

Write a Matlab function called `Dice(k)` that will open a new figure and display dice face k (from 1 to 6). An example for k = 3 is shown below. The challenge is to write the shortest possible code for the function. The length of the code is the number of characters ignoring the white spaces and new lines. (In real competitions, the variable names of any length are counted as one character, and comments are not counted at all. In our competition everything counts.)

```matlab
function Dice(k)
    figure
    hold
    rectangle('Po',[0 0 4 4],'Cu',.2,'Fa','k')
    b = [16 68 84 325 341 455]
    spy(reshape(str2num(dec2bin(b(k),9)'),3,3),'w',99)
    axis off
```

*Answer behind this shape*