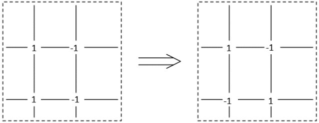
Introduction to Machine Learning

Designing Codebooks and ECOC



Learning goals

- Know what a codebook is
- Understand that codebooks generalize one-vs-one and one-vs-rest
- Know how to define a good codebook and error-correcting output codes (ECOC)
- Know how randomized hill-clmibing algorithm is used to find good codebooks

Designing Codebooks

CODEBOOKS

- We have already seen that we can write down principles like one-vs-rest and one-vs-one reduction compactly by so-called **codebooks**.
- During training, a scoring classifier is trained for each column.
- The k-th row is called **codeword** for class k.
- Knowing the principle of **codebooks**, we can define multiclass-to-binary reductions quite flexibly.
- We can now ask ourselves, how to create optimal codebooks.

Class	$f_1(\mathbf{x})$	$f_2(\mathbf{x})$	$f_3(\mathbf{x})$
1	1	-1	-1
2	-1	1	-1
3	- 1	-1	1

Class	$f_1(\mathbf{x})$	$f_2(\mathbf{x})$	$f_3(\mathbf{x})$
1	1	-1	0
2	-1	0	1
3	0	1	-1

Left: one-vs-rest codebook. Right: one-vs-one codebook.

CODEBOOKS: DECIDING LABELS

For a general codebook, once we trained the classifiers, how to predict the class $\hat{y}$ for a new input $\mathbf{x}$?

- When a new sample $\mathbf{x}$ is going to be classified, all classifiers f_k are applied to $\mathbf{x}$, scores are potentially transformed and turned into binary labels by $\text{sgn}\left(f_k(\mathbf{x})\right)$.

Class	$f_1(\mathbf{x})$	$f_2(\mathbf{x})$	$f_3(\mathbf{x})$
1	1	1	0
2	-1	1	1
3	0	-1	-1
sgn($\hat{f}(\mathbf{x})$)	**-1**	**1**	**- 1**

- We obtain a code for the observation $\mathbf{x}$ for which we can calculate the distance to the codewords of the other classes. This can be done by **Hamming distance** (counting the number of bits that differ) or by L_1-distance.

CODEBOOKS: DECIDING LABELS

- For example, the L_1-distance between $\operatorname{sgn}\left(\hat{f}(\mathbf{x})\right) = (-1, 1, -1)$ and the class 1 codeword $(1, 1, 0)$ is 3.
- We can do so for all the classes to obtain respective distances:

Classes	Dist
1	3
2	2
3	3

The distance for class 2 is minimal, therefore we predict class 2 for the input **x**.

DEFINING GOOD CODEBOOKS

Question: How to define a good codebook?

- Assume we are given a test observation $(\mathbf{x}, y)$ with $y = 2$.
- Assume classifier $f_2(\mathbf{x})$ produces a false prediction:

Class	$f_1(\mathbf{x})$	$f_2(\mathbf{x})$	$f_3(\mathbf{x})$	Dist
1	1	-1	0	3
2	-1	1	1	**2**
3	0	-1	-1	3
$\lvert\hat{\boldsymbol{f}}(\mathbf{x})\rvert$	**-1**	**-1**	**1**	

- Even though $f_2(\mathbf{x})$ is wrong, the overall prediction will be correct in the above case, if we pick the best codeword w.r.t. distance from the predicted codeword.
- We effectively **corrected** for the error.
- This motivates a desirable characteristic of a codebook: we want to have codes that can correct for as many errors as possible.
- Which is called **error-correcting output codes** (ECOC).

Error-Correcting Codes (ECOC)

ERROR-CORRECTING CODES (ECOC)

The power of a code to correct errors is related to the **row separation**:

- Each codeword should be well-separated in Hamming distance from each of the other codewords.
- Otherwise, if the class codewords are very similar, a prediction error in a single binary classifier easily results in an “overall” error.
- If the minimum distance between any pair of codewords is *d*, the code can correct at least $\left\lfloor \frac{d-1}{2} \right\rfloor$ single bit errors.

ERROR-CORRECTING CODES (ECOC)

Another desirable property is **column separation**:

- Columns should be uncorrelated.
- If two columns k and l are similar or identical, a learning algorithm will make similar (correlated) mistakes in learning f_k and f_l.
- Error-correcting codes only succeed if the errors made in the individual classifiers are relatively uncorrelated, so that the number of simultaneous errors in many classifiers is small.
- Errors in classifiers f_k and f_l will also be highly correlated if the bits in those columns are complementary.
- Try to ensure that columns are neither identical nor complementary.

$\rightarrow$ **We want to maximize distances between rows, and want the distances between columns to not be too small (identical columns) or too high (complementary columns).**

ERROR-CORRECTING CODES (ECOC)

Remark:

- In general, if there are k classes, there will be at most $2^{k-1} - 1$ usable binary columns.
- For example for $k = 3$, there are only $2^3 = 8$ possible columns. Of these, half are complements of the other half. The columns that only contain 1s or the one that only contains $-1s$ are also not usable.

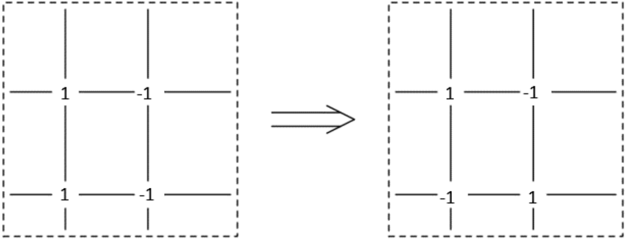
Class	$f_1(\mathbf{x})$	$f_2(\mathbf{x})$	$f_3(\mathbf{x})$	$f_4(\mathbf{x})$	$f_5(\mathbf{x})$	$f_6(\mathbf{x})$	$f_7(\mathbf{x})$	$f_8(\mathbf{x})$
1	-1	-1	-1	-1	1	1	1	1
2	-1	- 1	1	1	-1	-1	1	1
3	-1	1	-1	1	-1	1	-1	1

ERROR-CORRECTING CODES (ECOC)

Assume we have the budget to train L binary classifiers and now want to find an error-correcting code with maximal row and column separation.

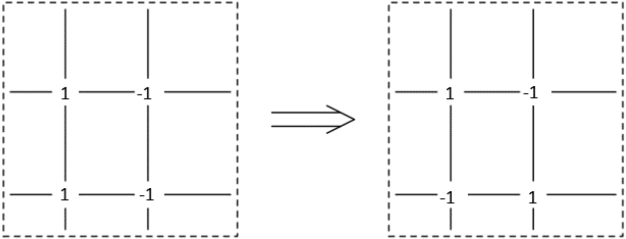
- For only few classes $g \leq 11$, exhaustive search can be performed and a codebook that has good row and column separation is chosen.
- However, for many classes $g > 11$, it becomes more and more challenging to find the optimal codebook with codewords of length L.
- *Dietterich et al.* employed a randomized hill-climbing algorithm for this task.

ECOC: RANDOMIZED HILL-CLIMBING ALGORITHM



- *g* codewords of length *L* are randomly drawn.
- Any pair of such random strings will be separated by a Hamming distance that is binomially distributed with mean $\frac{L}{2}$.
- The algorithm now iteratively improves the code: The algorithm repeatedly finds the pair of rows closest together (in Hamming distance or any other distance) and the pair of columns that have the “most extreme” distance (i.e. too close, or too far apart).

ECOC: RANDOMIZED HILL-CLIMBING ALGORITHM



- The algorithm then computes the four codeword bits where these rows and columns intersect and changes them to improve the row and column separations
- When the procedure reaches a local maximum, the algorithm randomly chooses pairs of rows and columns and tries to improve their separation.