Reducing Multiclass to Binary

LING572
Fei Xia
Week 7: 02/19/2010

Highlights

- What?
  - Converting a k-class problem to a binary problem.
- Why?
  - For some ML algorithms, a direct extension to the multiclass case may be problematic.
  - Ex: Boosting, support-vector machines (SVM)
- How?
  - Many methods

Methods

- One-vs-all
- All-pairs
- Error-correcting Output Codes (ECOC)**: see additional slides
- …

One-vs-all

- Idea:
  - Each class is compared to all others.
  - K classifiers: one classifier for each class.

- Training time:
  - For each class c_m, train a classifier $cl_m(x)$
    - replace (x,y) with

      (x, 1) if $y = c_m$

      (x, -1) if $y \neq c_m$

An example: training

- x1 c1 …
- x2 c2 …
- x3 c1 …
- x4 c3 …

for c1-vs-all:

x1 1 …

x2 -1 …

x3 1 …

x4 -1 …

for c2-vs-all:

x1 -1

x2 1 …

x3 -1 …

x4 -1 …

for c3-vs-all:

x1 -1…

x2 -1…

x3 -1 …

x4 1 …

One-vs-all (cont)

- Testing time: given a new example x
  - Run each of the k classifiers on x

  - Choose the class c_m with the highest confidence score $cl_m(x)$:

$$c^* = \arg\max_m cl_m(x)$$

An example: testing

- x1 c1 …
- x2 c2 …
- x3 c1 …
- x4 c3 …

➔ three classifiers

Test data:

x ?? f1 v1 …

for c1-vs-all:

x ?? 1 0.7 -1 0.3

for c2-vs-all

x ?? 1 0.2 -1 0.8

for c3-vs-all

x ?? 1 0.6 -1 0.4

=> what's the system prediction for x?

All-pairs

- Idea:
  - all pairs of classes are compared to each other
  - C_k^2 classifiers: one classifier for each class pair.

- Training:
  - For each pair (c_m, c_n) of classes, train a classifier cl_{mn}
    - replace a training instance (x,y) with

      $(x, 1)$ if $y = c_m$

      $(x, -1)$ if $y = c_n$

      otherwise ignore the instance

An example: training

- x1 c1 …
- x2 c2 …
- x3 c1 …
- x4 c3 …

for c1-vs-c2:

x1 1 …

x2 -1 …

x3 1 …

for c2-vs-c3:

x2 1 …

x4 -1 …

for c1-vs-c3:

x1 1…

x3 1 …

x4 -1 …

All-pairs (cont)

- Testing time: given a new example x
  - Run each of the C_k^2 classifiers on x
  - Max-win strategy: Choose the class c_m that wins the most pairwise comparisons:
  - Other coupling models have been proposed: e.g., (Hastie and Tibshirani, 1998)

An example: testing

- x1 c1 …
- x2 c2 …
- x3 c1 …
- x4 c3 …

➔ three classifiers

Test data:

x ?? f1 v1 …

for c1-vs-c2:

x ?? 1 0.7 -1 0.3

for c2-vs-c3

x ?? 1 0.2 -1 0.8

for c1-vs-c3

x ?? 1 0.6 -1 0.4

=> what's the system prediction for x?

Summary

- Different methods:
  - Direct multiclass
  - One-vs-all (a.k.a. one-per-class): k-classifiers
  - All-pairs: C_k^2 classifiers
  - ECOC: n classifiers (n is the num of columns)

- Some studies report that All-pairs and ECOC work better than one-vs-all.

Hw7

Q1: one-vs-all

- q1.sh training_data test_data output_dir > acc_file
- training_data, test_data: Mallet text format
  - instanceName goldClass f1 v1 f2 v2 …
- output_dir/: see the next slide
- acc_file: same as the one in hw2-hw5

output_dir/

- class_map: classname class_index
  - gun 1 \n mid-east 2 \n misc 3 \n

- 1-vs-all/, 2-vs-all/, 3-vs-all/
  - “train” and “test”: like training_data and test_data, but goldClass is changed to 1 or -1
  - “sys_output”:
    - instanceName goldClass c1 p1 c2 p2
    - c1 and c2 are 1 and -1 (the order depends on prob)
    - $p_i = P(c_i \mid x)$

- final_sys_out: “instName goldClassName [cn1 prob1 cn2 prob2 …]”
  - goldClassName and cn_i are classnames, not class index or 1, -1
  - prob_i is $P(1 \mid x)$ based the classifier i-vs-all
  - (cn_i, prob_i) pairs are sorted according to prob_i in descending order

Q2: all-pairs

- q2.sh training_data test_data output_dir > acc_file

- The file formats are the same except for the following under output-dir/:
  - 1-vs-2, 1-vs-3, 2-vs-3, not 1-vs-all, 2-vs-all, …
  - final_sys_output:
    - InstanceName goldClass sysClass 1:2 p(1,2) 1:3 p(1,3) … c1=n1 c2=n2 c3=n3
    - sysClass is the class name that q2.sh predicts for the instance
    - p(i,j) is the P(class=1 | x) based on i-vs-j classifier
    - n_i is the number of times that c_i wins

Additional slides

Error-correcting output codes (ECOC)

- Proposed by (Dietterich and Bakiri, 1995)

- Idea:
  - Each class is assigned a unique binary string of length n.
  - Train n classifiers, one for each bit.
  - Testing time: run n classifiers on x to get a n-bit string s, and choose the class which is closest to s.

An example

Class	Code Word					
	vl	hl	dl	cc	ol	or
0	0	0	0	1	0	0
1	1	0	0	0	0	0
2	0	1	1	0	1	0
3	0	0	0	0	1	0
4	1	1	0	0	0	0
5	1	1	0	0	1	0
6	0	0	1	1	0	1
7	0	0	1	0	0	0
8	0	0	0	1	0	0
9	0	0	1	1	0	0

Meaning of each column

Column position	Abbreviation	Meaning
1	vl	contains vertical line
2	hl	contains horizontal line
3	dl	contains diagonal line
4	cc	contains closed curve
5	ol	contains curve open to left
6	or	contains curve open to right

Another example: 15-bit code for a 10-class problem

Class	Code Word														
	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}
0	1	1	0	0	0	0	1	0	1	0	0	1	1	0	1
1	0	0	1	1	1	1	0	1	0	1	1	0	0	1	0
2	1	0	0	1	0	0	0	1	1	1	1	0	1	0	1
3	0	0	1	1	0	1	1	1	0	0	0	0	1	0	1
4	1	1	1	0	1	0	1	1	0	0	1	0	0	0	1
5	0	1	0	0	1	1	0	1	1	1	0	0	0	0	1
6	1	0	1	1	1	0	0	0	0	1	0	1	0	0	1
7	0	0	0	1	1	1	1	0	1	0	1	1	0	0	1
8	1	1	0	1	0	1	1	0	0	1	0	0	0	1	1
9	0	1	1	1	0	0	0	0	1	0	1	0	0	1	1

Hamming distance

- Definition: the **Hamming distance** between two strings of equal length is the number of positions for which the corresponding symbols are different.

- Ex:
  - 10111 and 10010
  - 2143 and 2233
  - Toned and roses

How to choose a good error-correcting code?

- Choose the one with large minimum Hamming distance between any pair of code words.

- If the min Hamming distance is d, then the code can correct at least (d-1)/2 single bit errors.

Two properties of a good ECOC

- Row separations: Each codeword should be well-separated in Hamming distance from each of the other codewords

- Column separation: Each bit-position function f_i should be uncorrelated with each of the other f_j.

All possible columns for a three-class problem

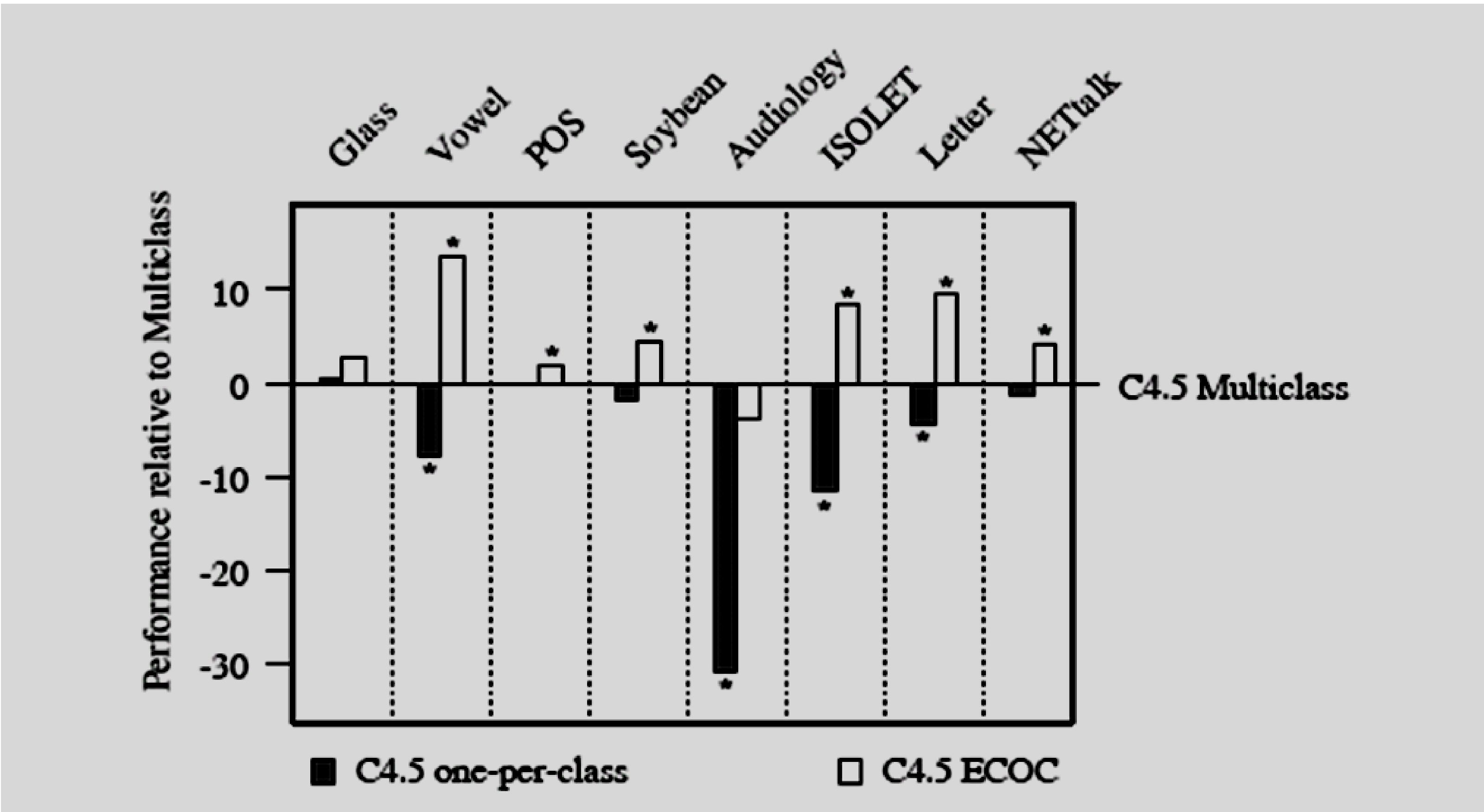
Class	Code Word							
	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7
c_0	0	0	0	0	1	1	1	1
c_1	0	0	1	1	0	0	1	1
c_2	0	1	0	1	0	1	0	1

If there are k classes, there will be at most 2^{k-1} -1 usable columns after removing complements and the all-zeros or all-ones column.

Finding a good code for different values of k

- Exhaustive codes
- Column selection from exhaustive codes
- Randomized hill climbing
- BCH codes
- …

Results



Glass
Vowel
POS
Soybean
Audiology
ISOLET
Letter
NETtalk
Performance relative to Multiclass
10
0
-10
-20
-30
C4.5 Multiclass
C4.5 one-per-class
C4.5 ECOC