Multi-Class Classification

Advanced Statistical Methods in NLP
Ling 572
February 14, 2012

Roadmap

- Motivation:
  - Binary and Multi-class: problems and classifiers
- Solving Multi-class problems with binary classifiers
  - One-vs-all
  - All-pairs
  - Error correcting output codes (ECOC) – overview

Classification Problems

- Some are naturally binary:

Classification Problems

- Some are naturally binary:
  - Spam tagging: Spam vs not-Spam

Classification Problems

- Some are naturally binary:
  - Spam tagging: Spam vs not-Spam
  - Segmentation tasks: Boundary vs Non-boundary

Classification Problems

- Some are naturally binary:
  - Spam tagging: Spam vs not-Spam
  - Segmentation tasks: Boundary vs Non-boundary
    - X1 X2 X3 X4 X5 X6 X7

Classification Problems

- Some are naturally binary:
  - Spam tagging: Spam vs not-Spam
  - Segmentation tasks: Boundary vs Non-boundary
    - X1 X2 X3 X4 X5 X6 X7
    - X1 X2 b X3 X4 X5 b X6 X7 b
    - Word, Sentence, topic, story

Classification Problems

- Some are naturally binary:
  - Spam tagging: Spam vs not-Spam
  - Segmentation tasks: Boundary vs Non-boundary
    - X1 X2 X3 X4 X5 X6 X7
    - X1 X2 b X3 X4 X5 b X6 X7 b
    - Word, Sentence, topic, story
  - Coreference:
    - Are two entities coreferent?

Classification Problems

- Many (most?) are multi-class

Classification Problems

- Many (most?) are multi-class
  - Most text classification:

Classification Problems

- Many (most?) are multi-class
  - Most text classification:
    - e.g. guns vs mideast vs misc

Classification Problems

- Many (most?) are multi-class
  - Most text classification:
    - e.g. guns vs mideast vs misc
  - Part-of-Speech tagging:

Classification Problems

- Many (most?) are multi-class
  - Most text classification:
    - e.g. guns vs mideast vs misc
  - Part-of-Speech tagging:
    - NN vs NNP vs VBZ vs RB vs DT vs.....

Classification Problems

- Many (most?) are multi-class
  - Most text classification:
    - e.g. guns vs mideast vs misc
  - Part-of-Speech tagging:
    - NN vs NNP vs VBZ vs RB vs DT vs.....
  - Named Entity Extraction:

Classification Problems

- Many (most?) are multi-class
  - Most text classification:
    - e.g. guns vs mideast vs misc
  - Part-of-Speech tagging:
    - NN vs NNP vs VBZ vs RB vs DT vs.....
  - Named Entity Extraction:
    - B-PER, I-PER, B-ORG, I-ORG, O,....
  - etc

Classifiers

- Also, binary or multi-class

Classifiers

- Also, binary or multi-class
- Many so far are directly multi-class
  - Specifically, can output more than two class labels

Classifiers

- Also, binary or multi-class
- Many so far are directly multi-class
  - Specifically, can output more than two class labels
  - Decision trees, Naïve Bayes, MaxEnt

Classifiers

- Also, binary or multi-class
- Many so far are directly multi-class
  - Specifically, can output more than two class labels
  - Decision trees, Naïve Bayes, MaxEnt
- Many other useful classifiers are basically binary
  - Perceptrons
  - Neural Networks
  - Support Vector Machines (next)

Binary & Multiclass: Classification & Classifiers

- If some classifiers are basically binary,
  - Does that mean we can only use them on binary tasks?

Binary & Multiclass: Classification & Classifiers

- If some classifiers are basically binary,
  - Does that mean we can only use them on binary tasks?
- No!

Binary & Multiclass: Classification & Classifiers

- If some classifiers are basically binary,
  - Does that mean we can only use them on binary tasks?
- No!
  - Otherwise this would be a very short class....

Binary & Multiclass: Classification & Classifiers

- If some classifiers are basically binary,
  - Does that mean we can only use them on binary tasks?
- No!
  - Otherwise this would be a very short class....
- Basic idea:
  - Decompose multi-class tasks into set of binary tasks
  - Create ensemble of binary classifiers for binary tasks
  - Combine outputs of ensemble as multi-class classifier

Questions & Approaches

- Questions:
  - How do we represent multi-class task in binary form?

Questions & Approaches

- Questions:
  - How do we represent multi-class task in binary form?
  - How do we integrate the outputs of binary classification for multiclass output?

Questions & Approaches

- Questions:
  - How do we represent multi-class task in binary form?
  - How do we integrate the outputs of binary classification for multiclass output?
- Approaches:
  - Correspond to different decompositions/integrations
  - One-vs-all
  - All-pairs
  - Error-correcting Output Codes (ECOC)

Multi-class via 1-vs-All

- Basic idea:
  - Which binary classifiers?

Multi-class via 1-vs-All

- Basic idea:
  - Which binary classifiers?
    - Instead of a single classifier with multiple outputs
    - Create classifiers that distinguish each class from all others

Multi-class via 1-vs-All

- Basic idea:
  - Which binary classifiers?
    - Instead of a single classifier with multiple outputs
    - Create classifiers that distinguish each class from all others
      - E.g. for POS tagging: DT vs not-DT, NN vs not-NN, etc
  - Combined how?

Multi-class via 1-vs-All

- Basic idea:
  - Which binary classifiers?
    - Instead of a single classifier with multiple outputs
    - Create classifiers that distinguish each class from all others
      - E.g. for POS tagging: DT vs not-DT, NN vs not-NN, etc
  - Combined how?
    - For each instance, run all classifiers
    - Return classifier with highest confidence/score

Training

- Create training data for 1-vs-all classifiers
- How many 1-vs-all classifiers?

Training

- Create training data for 1-vs-all classifiers
- How many 1-vs-all classifiers?
  - 1 per class: k-classes → k binary classifiers
- How do we map from multi-class training to binary?

Training

- Create training data for 1-vs-all classifiers
- How many 1-vs-all classifiers?
  - 1 per class: k-classes → k binary classifiers
- How do we map from multi-class training to binary?
  - For each class c_m,
    - For each training instance (x,y)
      - if $y=c_m$, create instance (x,1)
      - otherwise, create instance (x,-1)

Example: Training

- Original Data:
  - x1 c1 ...
  - x2 c3 ....
  - x3 c1 ....
  - x4 c2 ...
- 1-vs-all Training Data:

Example: Training

- Original Data:
  - x1 c1 ...
  - x2 c3 ....
  - x3 c1 ....
  - x4 c2 ...
- 1-vs-all Training Data:
- c1-vs-all:

Example: Training

- Original Data:
  - x1 c1 ...
  - x2 c3 ....
  - x3 c1 ....
  - x4 c2 ...
- 1-vs-all Training Data:
- c1-vs-all:
  - x1 1 ....
  - x2 -1 ...
  - x3 1 ....
  - x4 -1 .....
- c2-vs-all:

Example: Training

- Original Data:
  - x1 c1 ...
  - x2 c3 ....
  - x3 c1 ....
  - x4 c2 ...
- 1-vs-all Training Data:
- c1-vs-all:
  - x1 1 ....
  - x2 -1 ...
  - x3 1 ....
  - x4 -1 .....
- c2-vs-all:
  - x1 -1 ....
  - x2 -1 ....
  - x3 -1 ....
  - x4 1 ....
- c3-vs-all:
  - x1 -1 ....
  - x2 1 ....
  - x3 -1 ....
  - x4 -1 ...

Testing Example

- For each testing instance x,
  - Classify using all classifiers
  - Select
    - class $c^* = \text{argmax}_m \, cl_m(x)$

Testing Example

- For each testing instance x,
  - Classify using all classifiers
  - Select
    - class $c^* = \text{argmax}_m \, cl_m(x)$
- Consider example x
- Classifier c1-vs-all:
  - x 1 0.7 -1 0.3
- Classifier c2-vs-all:
  - x 1 0.2 -1 0.8
- Classifier c3-vs-all:
  - x 1 0.6 -1 0.4
- x?

All-pairs

- Basic idea:
  - Which binary classifiers?
    - Instead of a single classifier with multiple outputs
    - Create classifiers that distinguish each pair of classes

All-pairs

- Basic idea:
  - Which binary classifiers?
    - Instead of a single classifier with multiple outputs
    - Create classifiers that distinguish each pair of classes
      - e.g. POS: DT vs NN; DT vs RB; DT vs JJ; DT vs VBZ;...
  - Combined how?
    - For each instance, run all classifiers

All-pairs

- Basic idea:
  - Which binary classifiers?
    - Instead of a single classifier with multiple outputs
    - Create classifiers that distinguish each pair of classes
      - e.g. POS: DT vs NN; DT vs RB; DT vs JJ; DT vs VBZ;...
  - Combined how?
    - For each instance, run all classifiers
    - Return most frequent classification label

Training

- Create training data for all-pairs classifiers
- How many all-pairs classifiers?

Training

- Create training data for all-pairs classifiers
- How many all-pairs classifiers?
  - k classes → $O(k^2)$ binary classifiers
    - $k(k-1)/2$ actually

Training

- Create training data for all-pairs classifiers
- How many all-pairs classifiers?
  - k classes → $O(k^2)$ binary classifiers
    - $k(k-1)/2$ actually
- How do we map from multi-class training to binary?

Training

- Create training data for all-pairs classifiers
- How many all-pairs classifiers?
  - k classes → $O(k^2)$ binary classifiers
    - $k(k-1)/2$ actually
- How do we map from multi-class training to binary?
  - For each class c_i,
    - For each class c_j, $i<j<k$,
      - for each instance (x,y)
        - if $y=c_i$, create instance (x,1)
        - if $y=c_j$, create instance (x,-1),
        - o.w. ignore

Example: Training

- Original Data:
  - x1 c1 ...
  - x2 c3 ....
  - x3 c1 ....
  - x4 c2 ...
- All-pairs Training Data:
- c1-vs-c2:

Example: Training

- Original Data:
  - x1 c1 ...
  - x2 c3 ....
  - x3 c1 ....
  - x4 c2 ...
- All-pairs Training Data:
- c1-vs-c2:
  - x1 1 ....
  - x3 1 ....
  - x4 -1 .....
- c1-vs-c3:
- c2-vs-c3:

Example: Training

- Original Data:
  - x1 c1 ...
  - x2 c3 ....
  - x3 c1 ....
  - x4 c2 ...
- All-pairs Training Data:
- c1-vs-c2:
  - x1 1 ....
  - x3 1 ....
  - x4 -1 .....
- c1-vs-c3:
  - x1 1 ....
  - x2 -1 ....
  - x3 1 ....
- c2-vs-c3:

Example: Training

- Original Data:
  - x1 c1 ...
  - x2 c3 ....
  - x3 c1 ....
  - x4 c2 ...
- All-pairs Training Data:
- c1-vs-c2:
  - x1 1 ....
  - x3 1 ....
  - x4 -1 .....
- c1-vs-c3:
  - x1 1 ....
  - x2 -1 ....
  - x3 1 ....
- c2-vs-c3:
  - x2 -1 ....
  - x4 1 ...

Testing Example

- For each testing instance x,
  - Classify using all classifiers
  - Select
    - class c with most votes
  - Other variants

- Consider example x

Testing Example

- For each testing instance x,
  - Classify using all classifiers
  - Select
    - class c with most votes
  - Other variants

- Consider example x

- Classifier c1-vs-c2:
  - x 1 0.7 -1 0.3

- Classifier c2-vs-c3:
  - x 1 0.2 -1 0.8

- Classifier c1-vs-c3:
  - x 1 0.6 -1 0.4

- x?

Error-Correcting Output Codes

- Dietterich & Bakiri, 1995

- Basic idea:
  - Each class assigned a binary string of length *n* (codeword)

Error-Correcting Output Codes

- Dietterich & Bakiri, 1995

- Basic idea:
  - Each class assigned a binary string of length *n* (codeword)
  - Each bit position corresponds to output of classifier

Error-Correcting Output Codes

- Dietterich & Bakiri, 1995

- Basic idea:
  - Each class assigned a binary string of length *n* (codeword)
  - Each bit position corresponds to output of classifier
  - Training: train 1 classifier per bit position

Error-Correcting Output Codes

- Dietterich & Bakiri, 1995

- Basic idea:
  - Each class assigned a binary string of length *n* (codeword)
  - Each bit position corresponds to output of classifier
  - Training: train 1 classifier per bit position
  - Testing: apply each classifier to compute new codeword
    - Assign class with closest codeword

Example: Digit Recognition

- 6-bit code for 10-class problem
- 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

- Each row:
  - Codeword for class/digit

Direct Codes for Digit Recognition

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

ECOC for Digit Recognition

- Error correcting code for digit recognition
  - 15-bit code for 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

Decoding

- Decoding:
  - Label test instance with class with “closest” codeword

Decoding

- Decoding:
  - Label test instance with class with “closest” codeword
- What’s “closest”?

Decoding

- Decoding:
  - Label test instance with class with “closest” codeword
- What’s “closest”?
  - Many distances: Euclidean, cosine, Manhattan, etc
- Here, Hamming distance:

Decoding

- Decoding:
  - Label test instance with class with “closest” codeword
- What’s “closest”?
  - Many distances: Euclidean, cosine, Manhattan, etc
- Here, Hamming distance:
  - Count of number of bits that differ
    - E.g. 110001 maps to 110000

Decoding

- Decoding:
  - Label test instance with class with “closest” codeword
- What’s “closest”?
  - Many distances: Euclidean, cosine, Manhattan, etc
- Here, Hamming distance:
  - Count of number of bits that differ
    - E.g. 110001 maps to 110000
    - Hamming distance = 1

Error Correcting Output Codes

- Intuition:
  - Output class ‘transmitted’ through a noisy channel
  - Transmit via: features, training data, learning alg.
  - Errors may be introduced due to:
    - limited training data, bad features, poor learning

Error Correcting Output Codes

- Intuition:
  - Output class ‘transmitted’ through a noisy channel
  - Transmit via: features, training data, learning alg.
  - Errors may be introduced due to:
    - limited training data, bad features, poor learning
- ‘Meaningful’ or class-based codes non-optimal

Error Correcting Output Codes

- Intuition:
  - Output class ‘transmitted’ through a noisy channel
  - Transmit via: features, training data, learning alg.
  - Errors may be introduced due to:
    - limited training data, bad features, poor learning
- ‘Meaningful’ or class-based codes non-optimal
- Error-correcting codes can recover from some bit errors

Error Correction

- Quality of ECC:
  - Minimum distance b/t pair of codewords
- Error correction:

Error Correction

- Quality of ECC:
  - Minimum distance b/t pair of codewords
- Error correction:
  - Minimum Hamming distance b/t codes: *d*

Error Correction

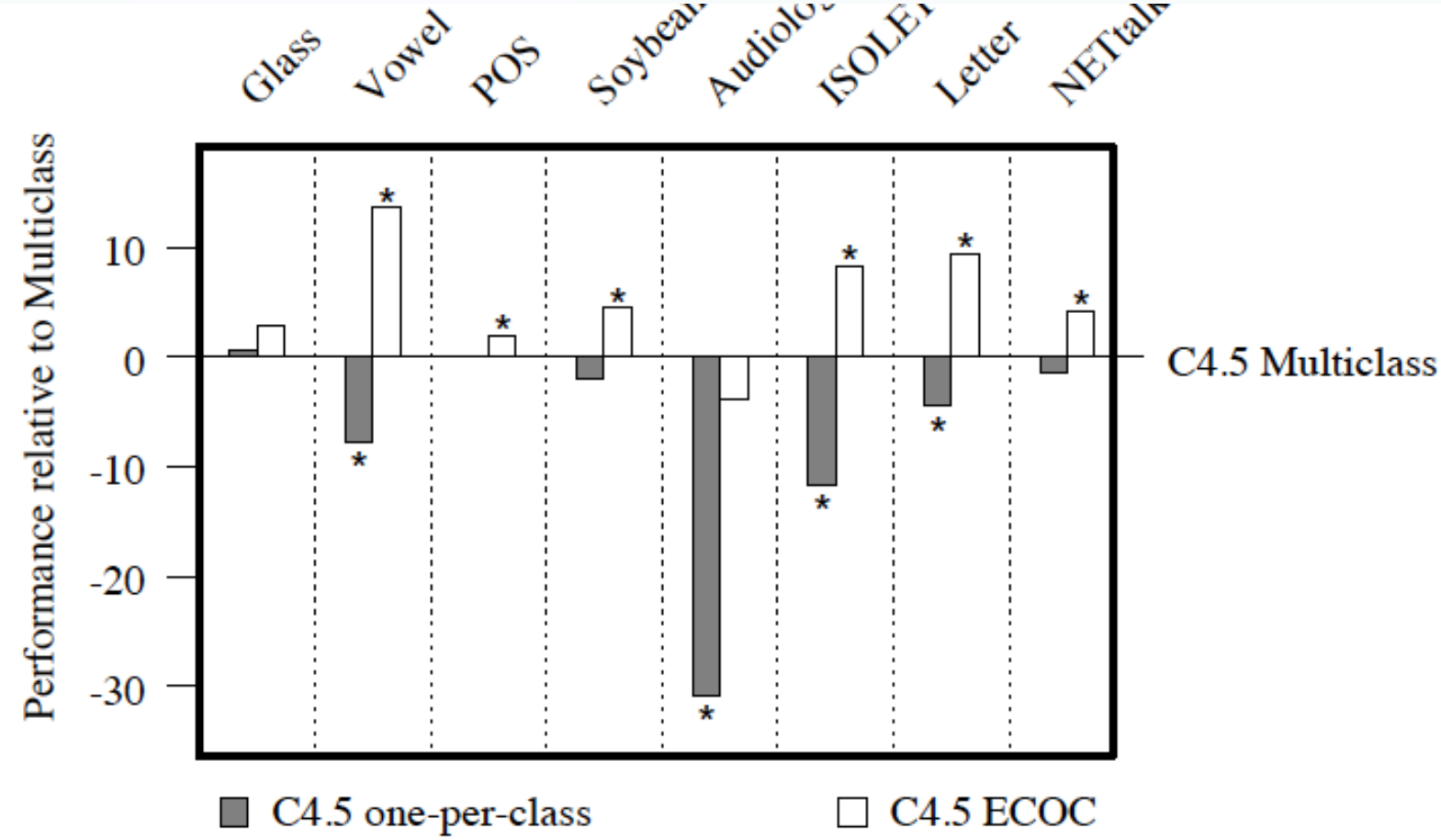
- Quality of ECC:
  - Minimum distance b/t pair of codewords
- Error correction:
  - Minimum Hamming distance b/t codes: *d*
  - # of correctable single bit errors: $\left\lfloor \frac{d-1}{2} \right\rfloor$

Error Correction

- Quality of ECC:
  - Minimum distance b/t pair of codewords
- Error correction:
  - Minimum Hamming distance b/t codes: *d*
  - # of correctable single bit errors: $\left\lfloor \frac{d-1}{2} \right\rfloor$
- 'Meaningful' digit codes: Min. distance = 1
  - No correction capacity

Comparison

- Direct multiclass, One-bit-per-class, ECOC
  - Decision trees



Creating Error Correcting Codes

- ECOC: Matrix
  - # columns: code length
  - # rows: # classes
  - Row = codeword

Creating Error Correcting Codes

- ECOC: Matrix
  - # columns: code length
  - # rows: # classes
  - Row = codeword
- Requirements for good codes:
  - Row separation:
    - Codewords well-separated in Hamming distance

Creating Error Correcting Codes

- ECOC: Matrix
  - # columns: code length
  - # rows: # classes
  - Row = codeword
- Requirements for good codes:
  - Row separation:
    - Codewords well-separated in Hamming distance
  - Column separation:
    - Columns should be uncorrelated with each other

Creating Error Correcting Codes

- ECOC: Matrix
  - # columns: code length
  - # rows: # classes
  - Row = codeword
- Requirements for good codes:
  - Row separation:
    - Codewords well-separated in Hamming distance
  - Column separation:
    - Columns should be uncorrelated with each other
    - Columns well-separated in Hamming distance
      - w.r.t. each other, and complement other columns
        - Complement b/c many classifiers symmetric

ECOC

- Tricky to create for < 5 classes

ECOC

- Tricky to create for < 5 classes
- With few classes limited # of distinct columns
  - [illegible]

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

ECOC

- Tricky to create for < 5 classes
- With few classes limited # of distinct columns
  - e.g. 3 classes, $2^3 = 8$ possible columns

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

  - Last 4 are complements of first 4
  - All same → non-discriminative
  - Only 3 distinct = # of classes

ECOC

- Tricky to create for < 5 classes
- With few classes limited # of distinct columns
  - e.g. 3 classes, $2^3 = 8$ possible columns

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

  - Last 4 are complements of first 4
  - All same → non-discriminative
  - Only 3 distinct = # of classes
  - for k classes: $2^{k-1}-1$ usable columns

Approaches for ECOC

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

Multi-classification Methods

- Approaches:
  - Direct multiclass
  - One-vs-all: k binary classifiers
  - All-pairs: $O(k^2)$ binary classifiers
  - ECOC: n binary classifiers (codeword length n)

Multi-classification Methods

- Approaches:
  - Direct multiclass
  - One-vs-all: k binary classifiers
  - All-pairs: $O(k^2)$ binary classifiers
  - ECOC: n binary classifiers (codeword length n)

- Effectiveness:
  - In experiments, all-pairs and ECOC often outperform one-vs-all