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)
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 $\rightarrow$ 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 $\rightarrow$ 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 $c^* = \arg\max_m c_l_m(x)$
For each testing instance $x$, classify using all classifiers and select
$$\text{class } c^* = \arg\max_m c_l_m(x)$$
Consider example $x$

- Classifier $c_1$-vs-all:
  - $x \ 1 \ 0.7 \ -1 \ 0.3$
- Classifier $c_2$-vs-all:
  - $x \ 1 \ 0.2 \ -1 \ 0.8$
- Classifier $c_3$-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 $\rightarrow O(k^2)$ binary classifiers
    - $k(k-1)/2$ actually
Training

- Create training data for all-pairs classifiers
- How many all-pairs classifiers?
  - $k$ classes $\Rightarrow O(k^2)$ 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 \( \rightarrow O(k^2) \) 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 ....
  - c2-vs-c3:
  - c1-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 \( c_{1}\text{-vs-}c_{2} \):
  - \( x \ 1 \ 0.7 \ -1 \ 0.3 \)

- Classifier \( c_{2}\text{-vs-}c_{3} \):
  - \( x \ 1 \ 0.2 \ -1 \ 0.8 \)

- Classifier \( c_{1}\text{-vs-}c_{3} \):
  - \( x \ 1 \ 0.6 \ -1 \ 0.4 \)

- \( x? \)
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:
  - \( \text{Binary function with meaning} \)

- Each row:
  - \( \text{Codeword for class/digit} \)

<table>
<thead>
<tr>
<th>Column position</th>
<th>Abbreviation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>vl</td>
<td>contains vertical line</td>
</tr>
<tr>
<td>2</td>
<td>hl</td>
<td>contains horizontal line</td>
</tr>
<tr>
<td>3</td>
<td>dl</td>
<td>contains diagonal line</td>
</tr>
<tr>
<td>4</td>
<td>cc</td>
<td>contains closed curve</td>
</tr>
<tr>
<td>5</td>
<td>ol</td>
<td>contains curve open to left</td>
</tr>
<tr>
<td>6</td>
<td>or</td>
<td>contains curve open to right</td>
</tr>
</tbody>
</table>
# Direct Codes for Digit Recognition

<table>
<thead>
<tr>
<th>Class</th>
<th>Code Word</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>vl</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
</tr>
</tbody>
</table>
ECOC for Digit Recognition

- Error correcting code for digit recognition
- 15-bit code for 10 class problem

<table>
<thead>
<tr>
<th>Class</th>
<th>$f_0$</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>
<th>$f_8$</th>
<th>$f_9$</th>
<th>$f_{10}$</th>
<th>$f_{11}$</th>
<th>$f_{12}$</th>
<th>$f_{13}$</th>
<th>$f_{14}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
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 between a pair of codewords

- Error correction:
  - Minimum Hamming distance between codes: $d$
Error Correction

- Quality of ECC:
  - Minimum distance between pair of codewords

- Error correction:
  - Minimum Hamming distance between codes: $d$
  - Number of correctable single bit errors: $\left\lfloor \frac{d - 1}{2} \right\rfloor$
Error Correction

- Quality of ECC:
  - Minimum distance between pair of codewords

- Error correction:
  - Minimum Hamming distance between codes: $d$
  - Number of correctable single bit errors: $\left\lfloor \frac{d-1}{2} \right\rfloor$
  - ‘Meaningful’ digit codes: Minimum 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

<table>
<thead>
<tr>
<th>Class</th>
<th>$f_0$</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>$c_0$</td>
<td>0</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>$c_1$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$c_2$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
ECOC

- Tricky to create for < 5 classes
- With few classes limited # of distinct columns
  - e.g. 3 classes: $2^3 = 8$ possible columns
  - Last 4 are complements of first 4
  - All same $\rightarrow$ 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
  - Last 4 are complements of first 4
  - All same $\rightarrow$ 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