Slides adapted from Prof Carpuat and Duraiswami

CMSC 422 Introduction to Machine Learning

Lecture 10 Multiclass Classification and Reductions

Furong Huang / furongh@cs.umd.edu

UNIVERSITY OF MARYLAND

Topics

Given an arbitrary method for binary classification, how can we learn to make multiclass predictions?

Fundamental ML concept: reductions

One Example of Reduction: Learning with Imbalanced Data

TASK: α-WEIGHTED BINARY CLASSIFICATION

Given:

1. An input space $\mathcal{X}$
2. An unknown distribution $\mathcal{D}$ over $\mathcal{X} \times \{-1, +1\}$

Compute: A function f minimizing: $\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\alpha^{y=1}\left[f(x) \neq y\right]\right]$

Subsampling Optimality Theorem:
If the binary classifier achieves a binary error rate of ε, then the error rate of the α-weighted classifier is α ε

Another Example of Reduction: Multiclass Classification

Task: Multiclass Classification

Given:

1. An input space $\mathcal{X}$ and number of classes K
2. An unknown distribution $\mathcal{D}$ over $\mathcal{X} \times [K]$

Compute: A function f minimizing: $\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[f(\boldsymbol{x}) \neq y\right]$

Reduction 1: OVA

- "One versus all" (aka "one versus rest")
  - Train K-many binary classifiers
  - classifier k predicts whether an example belong to class k or not
  - At test time,
    - If only one classifier predicts positive, predict that class
    - Break ties randomly

Algorithm 12 OneVersusAllTrain($\mathbf{D}^{multiclass}$, BinaryTrain)

1: **for** $i = 1$ **to** K **do**
2: $\quad \mathbf{D}^{bin} \leftarrow$ relabel $\mathbf{D}^{multiclass}$ so class i is positive and $\neg i$ is negative
3: $\quad f_i \leftarrow$ BinaryTrain($\mathbf{D}^{bin}$)
4: **end for**
5: **return** $f_1, \ldots, f_K$

Algorithm 13 OneVersusAllTest($f_1, \ldots, f_K, \hat{\boldsymbol{x}}$)

1: $score \leftarrow \langle 0, 0, \ldots, 0 \rangle$ // initialize K-many scores to zero
2: **for** $i = 1$ **to** K **do**
3: $\quad y \leftarrow f_i(\hat{\boldsymbol{x}})$
4: $\quad score_i \leftarrow score_i + y$
5: **end for**
6: **return** $\text{argmax}_k \; score_k$

Time complexity

- Suppose you have N training examples, in K classes. How long does it take to train an OVA classifier
  - if the base binary classifier takes O(N) time to learn?
  - if the base binary classifier takes O(N^2) time to learn?

Error bound

- **Theorem:** Suppose that the average error of the K binary classifiers is ε, then the error rate of the OVA multiclass classifier is at most (K-1) ε

- To prove this: how do different errors affect **the maximum ratio of the probability of a multiclass error to the number of binary errors** (“**efficiency**”)?

Error bound proof

1. If we have a **false negative** on one of the binary classifiers (assuming all other classifiers correctly output negative)

What is the probability that we will make an incorrect multiclass prediction?

$$(K - 1) / K$$

Efficiency: $[(K - 1) / K] / 1 = (K - 1) / K$

Error bound proof

2. If we have m **false positives** with the binary classifiers

What is the probability that we will make an incorrect multiclass prediction?

If there is also a false negative: 1

Efficiency =1 / (m + 1)

Otherwise m / (m + 1)

Efficiency = [m / (m + 1)] / m = 1 / (m + 1)

Error bound proof

3. What is the worst case scenario?

False negative case: efficiency is (K-1)/K

- Larger than false positive efficiencies

There are K-many opportunities to get false negative, **overall error bound is (K-1) ε**

Reduction 2: AVA

All versus all (aka all pairs)

How many binary classifiers does this require?

Algorithm 14 AllVersusAllTrain($\mathbf{D}^{multiclass}$, BinaryTrain)

$f_{ij} \leftarrow \emptyset, \forall 1 \leq i < j \leq K$
for $i = 1$ **to** K-1 **do**
  $\mathbf{D}^{pos} \leftarrow$ all $\boldsymbol{x} \in \mathbf{D}^{multiclass}$ labeled i
  for $j = i+1$ **to** K **do**
    $\mathbf{D}^{neg} \leftarrow$ all $\boldsymbol{x} \in \mathbf{D}^{multiclass}$ labeled j
    $\mathbf{D}^{bin} \leftarrow \{(\boldsymbol{x}, +1) : \boldsymbol{x} \in \mathbf{D}^{\text{pos}}\} \cup \{(\boldsymbol{x}, -1) : \boldsymbol{x} \in \mathbf{D}^{\text{neg}}\}$
    $f_{ij} \leftarrow$ BinaryTrain($\mathbf{D}^{bin}$)
  end for
end for
return all f_{ij}s

Algorithm 15 AllVersusAllTest(all f_{ij}, $\hat{\boldsymbol{x}}$)

$score \leftarrow \langle 0, 0, \ldots, 0 \rangle$ // initialize K-many scores to zero
for $i = 1$ **to** K-1 **do**
  for $j = i+1$ **to** K **do**
    $y \leftarrow f_{ij}(\hat{\boldsymbol{x}})$
    $score_i \leftarrow score_i + y$
    $score_j \leftarrow score_j - y$
  end for
end for
return $\text{argmax}_k \; score_k$

Time complexity

- Suppose you have N training examples, in K classes. How long does it take to train an AVA classifier
  - if the base binary classifier takes O(N) time to learn?
  - if the base binary classifier takes O(N^2) time to learn?

Error bound

Theorem: Suppose that the average error of the K binary classifiers is ε, then the error rate of the AVA multiclass classifier is at most 2(K-1) ε

Question: Does this mean that AVA is always worse than OVA?

Extensions

- Divide and conquer
  - Organize classes into binary tree structures
- Use confidence to weight predictions of binary classifiers
  - Instead of using majority vote

Topics

- Given an arbitrary method for binary classification, how can we learn to make multiclass predictions?
  - OVA, AVA
- Fundamental ML concept: reductions

Ranking

Canonical example: web search

Given all the documents on the web

For a user query, retrieve relevant documents, ranked from most relevant to least relevant

How can we reduce ranking to binary classification?

Preference function

- Given a query q and documents d_i and d_j, the preference function outputs whether
  - d_i should be preferred to d_j
  - Or d_j should be preferred to d_i
- That's a binary classification problem!

Specifying the reduction from ranking to binary classification

- How to train classifier that predicts preferences?
- How to turn the predicted preferences into a ranking?

Algorithm 16 NAIVERANKTRAIN(*RankingData*, BINARYTRAIN)

1: $\mathbf{D} \leftarrow [\,]$
2: **for** $n = 1$ **to** N **do**
3: **for all** $i, j = 1$ **to** M **and** $i \neq j$ **do**
4: **if** i is prefered to j on query n **then**
5: $\mathbf{D} \leftarrow \mathbf{D} \oplus (\boldsymbol{x}_{nij}, +1)$
6: **else if** j is prefered to i on query n **then**
7: $\mathbf{D} \leftarrow \mathbf{D} \oplus (\boldsymbol{x}_{nij}, -1)$
8: **end if**
9: **end for**
10: **end for**
11: **return** BINARYTRAIN($\mathbf{D}$)

Features associated with comparing document j and document j for query n

Algorithm 17 NAIVERANKTEST(f, $\hat{\boldsymbol{x}}$)

1: $score \leftarrow \langle 0, 0, \ldots, 0 \rangle$ // initialize M-many scores to zero
2: **for all** $i, j = 1$ **to** M **and** $i \neq j$ **do**
3: $y \leftarrow f(\hat{\boldsymbol{x}}_{ij})$ // get predicted ranking of i and j
4: $score_i \leftarrow score_i + y$
5: $score_j \leftarrow score_j - y$
6: **end for**
7: **return** ARGSORT($score$) // return queries sorted by score

Naïve approach

Works well for bipartite problems

- “is this document relevant or not?”

Not ideal for full ranking problems, because

- Binary preference problems are not all equally important
- Separates preference function and sorting

Improving on naïve approach

TASK: ω-RANKING

Given:

1. An input space $\mathcal{X}$
2. An unknown distribution $\mathcal{D}$ over $\mathcal{X} \times \Sigma_M$

Compute: A function $f : \mathcal{X} \to \Sigma_M$ minimizing:

$$\mathbb{E}_{(\boldsymbol{x},\sigma)\sim\mathcal{D}}\left[\sum_{u\neq v}[\sigma_u < \sigma_v]\,[\hat{\sigma}_v < \hat{\sigma}_u]\,\omega(\sigma_u,\sigma_v)\right] \tag{5.7}$$

where $\hat{\sigma} = f(\boldsymbol{x})$

Example of cost functions

$$\omega(i,j) = \begin{cases} 1 & \text{if } \min\{i,j\} \leq K \text{ and } i \neq j \\ 0 & \text{otherwise} \end{cases}$$

Resulting Ranking Algorithms

Algorithm 18 RANKTRAIN($\mathbf{D}^{rank}$, ω, BINARYTRAIN)

1: $\mathbf{D}^{bin} \leftarrow [\,]$
2: **for all** $(\boldsymbol{x}, \sigma) \in \mathbf{D}^{rank}$ **do**
3: **for all** $u \neq v$ **do**
4: $y \leftarrow \text{SIGN}(\sigma_v - \sigma_u)$ // y is +1 if u is prefered to v
5: $w \leftarrow \omega(\sigma_u, \sigma_v)$ // w is the cost of misclassification
6: $\mathbf{D}^{bin} \leftarrow \mathbf{D}^{bin} \oplus (y, w, \boldsymbol{x}_{uv})$
7: **end for**
8: **end for**
9: **return** BINARYTRAIN($\mathbf{D}^{bin}$)

Algorithm 19 RANKTEST(f, $\hat{\boldsymbol{x}}$, obj)

```
1:  if obj contains 0 or 1 elements then
2:      return obj
3:  else
4:      p ← randomly chosen object in obj                          // pick pivot
5:      left ← [ ]                                  // elements that seem smaller than p
6:      right ← [ ]                                  // elements that seem larger than p
7:      for all u ∈ obj \{p} do
8:          ŷ ← f(x_up)                  // what is the probability that u precedes p
9:          if uniform random variable < ŷ then
10:             left ← left ⊕ u
11:         else
12:             right ← right ⊕ u
13:         end if
14:     end for
15:     left ← RANKTEST(f, x̂, left)                          // sort earlier elements
16:     right ← RANKTEST(f, x̂, right)                         // sort later elements
17:     return left ⊕ ⟨p⟩ ⊕ right
18: end if
```

RankTest

A probabilistic version of the quicksort algorithm

Only $O(M\log_2 M)$ calls to f in expectation

Better error bound than naïve algorithm

(see CIML for theorem)

What you should know

- What are reductions and why they are useful
- Implement, analyze and prove error bounds of algorithms for
  - Weighted binary classification
  - Multiclass classification (OVA, AVA)
- Understand algorithms for
  - ω –ranking

UNIVERSITY OF
MARYLAND

Furong Huang
3251 A.V. Williams, College Park, MD 20740
301.405.8010 / furongh@cs.umd.edu