Beyond Binary Classification: Reductions

CMSC 422

MARINE CARPUAT
marine@cs.umd.edu

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

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

Multiclass classification

- Real world problems often have multiple classes (text, speech, image, biological sequences…)
- How can we perform multiclass classification?
  - Straightforward with decision trees or KNN
  - Can we use the perceptron algorithm?

Today: Reductions for 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: $\mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}\left[f(\boldsymbol{x}) \neq y\right]$

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

How many classes can we handle in practice?

- In most tasks, number of classes K < 100
- For much larger K
  - we need to frame the problem differently
  - e.g, machine translation or automatic speech recognition

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

```
for i = 1 to K do
    D^bin ← relabel D^multiclass so class i is positive and ¬i is negative
    f_i ← BinaryTrain(D^bin)
end for
return f_1, . . . , f_K
```

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

```
score ← ⟨0, 0, . . . , 0⟩                    // initialize K-many scores to zero
for i = 1 to K do
    y ← f_i(x̂)
    score_i ← score_i + y
end for
return 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

- 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

- If we have p **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 / (p + 1)
  - If there is no false negative: p / (p + 1)
    - Efficiency = p / (p + 1) / p = 1 / (p + 1)

Error bound proof

- 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 $\textsc{AllVersusAllTrain}(\mathbf{D}^{multiclass}, \textsc{BinaryTrain})$

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

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

$score \leftarrow \langle 0, 0, \dots, 0 \rangle$ // 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 di and dj, the preference function outputs whether
  - di should be preferred to dj
  - Or dj should be preferred to di
- That’s a binary classification problem!

Specifiying 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)

```
D ← [ ]
for n = 1 to N do
    for all i, j = 1 to M and i ≠ j do
        if i is prefered to j on query n then
            D ← D ⊕ (x_nij, +1)
        else if j is prefered to i on query n then
            D ← D ⊕ (x_nij, −1)
        end if
    end for
end for
return BinaryTrain(D)
```

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

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

```
score ← ⟨0, 0, . . . , 0⟩                    // initialize M-many scores to zero
for all i, j = 1 to M and i ≠ j do
    y ← f(x̂_ij)                              // get predicted ranking of i and j
    score_i ← score_i + y
    score_j ← score_j - y
end for
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)

$\mathbf{D}^{bin} \leftarrow [\ ]$
for all $(\boldsymbol{x}, \sigma) \in \mathbf{D}^{rank}$ **do**
  for all $u \neq v$ **do**
    $y \leftarrow$ sign$(\sigma_v - \sigma_u)$ // y is +1 if u is prefered to v
    $w \leftarrow \omega(\sigma_u, \sigma_v)$ // w is the cost of misclassification
    $\mathbf{D}^{bin} \leftarrow \mathbf{D}^{bin} \oplus (y, w, \boldsymbol{x}_{uv})$
  end for
end for
return BinaryTrain($\mathbf{D}^{bin}$)

Algorithm 19 $\textsc{RankTest}(f, \hat{x}, obj)$

if obj contains 0 or 1 elements **then**
    return obj
else
    $p \leftarrow$ randomly chosen object in obj // pick pivot
    $left \leftarrow [\,]$ // elements that seem smaller than p
    $right \leftarrow [\,]$ // elements that seem larger than p
    for all $u \in obj \setminus \{p\}$ **do**
        $\hat{y} \leftarrow f(x_{up})$ // what is the probability that u precedes p
        if uniform random variable $< \hat{y}$ **then**
            $left \leftarrow left \oplus u$
        else
            $right \leftarrow right \oplus u$
        end if
    end for
    $left \leftarrow \textsc{RankTest}(f, \hat{x}, left)$ // sort earlier elements
    $right \leftarrow \textsc{RankTest}(f, \hat{x}, right)$ // sort later elements
    return $left \oplus \langle p \rangle \oplus right$
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
  - $\omega -$ranking

*for weighted binary classification and OVA only