More Reductions: Collective Classification & Learning to Rank

CMSC 422

MARINE CARPUAT
marine@cs.umd.edu

Topics

- Fundamental ML concept: reductions
  - re-use simple and efficient classification algorithms to perform more complex tasks
- We've seen 2 examples of reductions already
  - Weighted binary classification
  - Multiclass classification
- Today: 2 new examples
  - Collective classification
  - Learning to rank

A taste of more complex problems: Collective Classification

- Prediction problems where we want to make **multiple correlated predictions** simultaneously

- Examples:
  - object detection in an image
  - finding part of speech of words in a sentence

Collective vs. multiclass classification

- We can encode dependencies/correlations
  - Between input features only in multiclass classification
  - Between output predictions in collective classification

TASK: COLLECTIVE CLASSIFICATION

Given:

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

Compute: A function $f : \mathcal{G}(\mathcal{X}) \rightarrow \mathcal{G}([K])$ minimizing: $\mathbb{E}_{(V,E)\sim\mathcal{D}}\left[\sum_{v\in V}\left[\hat{y}_v \neq y_v\right]\right]$, where y_v is the label associated with vertex v in G and $\hat{y}_v$ is the label predicted by $f(G)$.

How to solve collective classification?

- One possible solution: stacking
  - reduces to binary/multiclass classification
  - Learn a stack of K classifiers
    - Classifier at level k uses output of lower level classifiers (k-1, k-2, etc.)
- Many other solutions exist
  - but are out of scope for 422
  - e.g. structured prediction, graphical models, hidden markov models

Algorithm 20 StackTrain($\mathcal{D}^{cc}$, K, MulticlassTrain)

1: $\mathbf{D}^{mc} \leftarrow [\,]$ // our generated multiclass data
2: $\hat{Y}_{k,n,v} \leftarrow 0, \forall k \in [K], n \in [N], v \in G_n$ // initialize predictions for all levels
3: **for** $k = 1$ **to** K **do**
4: **for** $n = 1$ **to** N **do**
5: **for all** $v \in G_n$ **do**
6: $(\boldsymbol{x}, y) \leftarrow$ features and label for node v
7: $\boldsymbol{x} \leftarrow \boldsymbol{x} \oplus \hat{Y}_{l,n,u}, \forall u \in \mathcal{N}(u), \forall l \in [k-1]$ // add on features for
8: // neighboring nodes from lower levels in the stack
9: $\mathbf{D}^{mc} \leftarrow \mathbf{D}^{mc} \oplus (y, \boldsymbol{x})$ // add to multiclass data
10: **end for**
11: **end for**
12: $f_k \leftarrow$ MulticlassTrain($\mathbf{D}^{bin}$) // train kth level classifier
13: **for** $n = 1$ **to** N **do**
14: $\hat{Y}_{k,n,v} \leftarrow$ StackTest($f_1, \ldots, f_k, G_n$) // predict using kth level classifier
15: **end for**
16: **end for**
17: **return** $f_1, \ldots, f_K$ // return all classifiers

CIML TYPO: this should be N(v)

CIML TYPO: this should be D^{mc}

Algorithm 21 StackTest($f_1, \ldots, f_K, G$)

1: $\hat{Y}_{k,v} \leftarrow 0, \forall k \in [K], v \in G$ // initialize predictions for all levels
2: **for** $k = 1$ **to** K **do**
3: **for all** $v \in G$ **do**
4: $\boldsymbol{x} \leftarrow$ features for node v
5: $\boldsymbol{x} \leftarrow \boldsymbol{x} \oplus \hat{Y}_{l,u}, \forall u \in \mathcal{N}(u), \forall l \in [k-1]$ // add on features for
6: // neighboring nodes from lower levels in the stack
7: $\hat{Y}_{k,v} \leftarrow f_k(\boldsymbol{x})$ // predict according to kth level
8: **end for**
9: **end for**
10: **return** $\{\hat{Y}_{K,v} : v \in G\}$ // return predictions for every node from the last layer

CIML TYPO: this should be N(v)

Stacking issues

- Very sensitive to overfitting
  - If f_1 makes perfect predictions on the training set, the other classifiers don't have much to learn from
- Solutions
  - Cross-validation on the training data
  - Regularize base learner heavily

Topics

- Fundamental ML concept: reductions
  - re-use simple and efficient classification algorithms to perform more complex tasks
- We've seen 2 examples of reductions already
  - Weighted binary classification
  - Multiclass classification
- Today: 2 new examples
  - Collective classification
  - Learning to rank

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)

```
1:  D ← [ ]
2:  for n = 1 to N do
3:      for all i, j = 1 to M and i ≠ j do
4:          if i is prefered to j on query n then
5:              D ← D ⊕ (x_nij, +1)
6:          else if j is prefered to i on query n then
7:              D ← D ⊕ (x_nij, −1)
8:          end if
9:      end for
10: end for
11: return BINARYTRAIN(D)
```

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

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

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

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
  - $O(M\log_2 M)$ calls to f only
  - Better error bound

    (see CIML for theorem, and optionally proof)

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, tree)
- Understand algorithms for
  - Stacking for collective classification
  - ω –ranking