Slides adapted from Prof Carpuat and Duraiswami

CMSC 422 Introduction to Machine Learning

Lecture 13 Bias and Fairness

Furong Huang / furongh@umd.edu

Some ML issues in the real world

UNIVERSITY OF MARYLAND

Ranking (Optional)

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)

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

$\mathbf{D} \leftarrow [\,]$
for $n = 1$ **to** N **do**
  for all $i, j = 1$ **to** M **and** $i \neq j$ **do**
    if i is prefered to j on query n **then**
      $\mathbf{D} \leftarrow \mathbf{D} \oplus (x_{nij}, +1)$
    else if j is prefered to i on query n **then**
      $\mathbf{D} \leftarrow \mathbf{D} \oplus (x_{nij}, -1)$
    end if
  end for
end for
return BINARYTRAIN($\mathbf{D}$)

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

$score \leftarrow \langle 0, 0, \dots, 0 \rangle$ // initialize M-many scores to zero
for all $i, j = 1$ **to** M **and** $i \neq j$ **do**
  $y \leftarrow f(\hat{x}_{ij})$ // get predicted ranking of i and j
  $score_i \leftarrow score_i + y$
  $score_j \leftarrow score_j - y$
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

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

Resulting Ranking Algorithms

Algorithm 18 RankTrain($\mathbf{D}^{rank}$, ω, BinaryTrain)

```
D^bin ← [ ]
for all (x, σ) ∈ D^rank do
    for all u ≠ v do
        y ← SIGN(σ_v - σ_u)                      // y is +1 if u is prefered to v
        w ← ω(σ_u, σ_v)                          // w is the cost of misclassification
        D^bin ← D^bin ⊕ (y, w, x_uv)
    end for
end for
return BinaryTrain(D^bin)
```

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

Exercise: understand this offline

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
  - ω −ranking

Bias and Fairness

UNIVERSITY OF
MARYLAND

Word Embeddings Could be Biased

Man is to Computer Programmer as Woman is to Homemaker?
Debiasing Word Embeddings

Extreme *she* occupations

1. homemaker	2. nurse	3. receptionist
4. librarian	5. socialite	6. hairdresser
7. nanny	8. bookkeeper	9. stylist
10. housekeeper	11. interior designer	12. guidance counselor

Extreme *he* occupations

1. maestro	2. skipper	3. protege
4. philosopher	5. captain	6. architect
7. financier	8. warrior	9. broadcaster
10. magician	11. figher pilot	12. boss

Figure 1: The most extreme occupations as projected on to the *she*−*he* gender direction on g2vNEWS. Occupations such as *businesswoman*, where gender is suggested by the orthography, were excluded.

Bolukbasi et al. NIPS 2016.

Machine Bias

There's software used across the country to predict future criminals. And it's biased against blacks.

by Julia Angwin, Jeff Larson, Surya Mattu and Lauren Kirchner, ProPublica

May 23, 2016

Prediction Fails Differently for Black Defendants

	WHITE	AFRICAN AMERICAN
Labeled Higher Risk, But Didn't Re-Offend	23.5%	44.9%
Labeled Lower Risk, Yet Did Re-Offend	47.7%	28.0%

Overall, Northpointe's assessment tool correctly predicts recidivism 61 percent of the time. But blacks are almost twice as likely as whites to be labeled a higher risk but not actually re-offend. It makes the opposite mistake among whites: They are much more likely than blacks to be labeled lower risk but go on to commit other crimes. (Source: ProPublica analysis of data from Broward County, Fla.)

https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing

Recall: Formal Definition of Binary Classification (from CIML)

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

Train/Test Mismatch

- When working with real world data, training sample
  - reflects human biases
  - is influenced by practical concerns
    - e.g., what kind of data is easy to obtain
- Train/test distribution mismatch is frequent issue
  - aka sample selection bias, domain adaptation

Domain Adaptation

- What does it mean for 2 distributions to be related?
- When 2 distributions are related how can we build models that effectively share information between them?

Unsupervised adaptation

Goal: learn a classifier f that achieves low expected loss under new distribution $\mathcal{D}^{new}$

Given labeled training data from old distribution

$\mathcal{D}^{\text{old}}$ $(x_1, y_1), \ldots, (x_N, y_N)$

And unlabeled examples from new distribution

$\mathcal{D}^{\text{new}}: z_1, \ldots, z_M$

Relation between test loss in new domain and old domain

$$
\begin{aligned}
&\text{test loss} && (8.1)\\
&= \mathbb{E}_{(x,y)\sim\mathcal{D}^{\text{new}}}\left[\ell(y, f(x))\right] \quad \text{definition} && (8.2)\\
&= \sum_{(x,y)} \mathcal{D}^{\text{new}}(x,y)\,\ell(y, f(x)) \quad \text{expand expectation} && (8.3)\\
&= \sum_{(x,y)} \mathcal{D}^{\text{new}}(x,y)\,\frac{\mathcal{D}^{\text{old}}(x,y)}{\mathcal{D}^{\text{old}}(x,y)}\,\ell(y, f(x)) \quad \text{times one} && (8.4)\\
&= \sum_{(x,y)} \mathcal{D}^{\text{old}}(x,y)\,\frac{\mathcal{D}^{\text{new}}(x,y)}{\mathcal{D}^{\text{old}}(x,y)}\,\ell(y, f(x)) \quad \text{rearrange} && (8.5)\\
&= \mathbb{E}_{(x,y)\sim\mathcal{D}^{\text{old}}}\left[\frac{\mathcal{D}^{\text{new}}(x,y)}{\mathcal{D}^{\text{old}}(x,y)}\,\ell(y, f(x))\right] \quad \text{definition} && (8.6)
\end{aligned}
$$

How can we estimate the ratio between Dnew and Dold?

Fixed base distribution

S = selection variable

$$\frac{\mathcal{D}^{\text{new}}(\boldsymbol{x},y)}{\mathcal{D}^{\text{old}}(\boldsymbol{x},y)} = \frac{\frac{1}{Z^{\text{new}}}\mathcal{D}^{\text{base}}(\boldsymbol{x},y)p(s=0 \mid \boldsymbol{x})}{\frac{1}{Z^{\text{old}}}\mathcal{D}^{\text{base}}(\boldsymbol{x},y)p(s=1 \mid \boldsymbol{x})} \qquad \text{definition} \quad (8.9)$$

$$= \frac{\frac{1}{Z^{\text{new}}}p(s=0 \mid \boldsymbol{x})}{\frac{1}{Z^{\text{old}}}p(s=1 \mid \boldsymbol{x})} \qquad \text{cancel base} \quad (8.10)$$

$$= Z\frac{p(s=0 \mid \boldsymbol{x})}{p(s=1 \mid \boldsymbol{x})} \qquad \text{consolidate} \quad (8.11)$$

$$= Z\frac{1-p(s=1 \mid \boldsymbol{x})}{p(s=1 \mid \boldsymbol{x})} \qquad \text{binary selection} \quad (8.12)$$

$$= Z\left[\frac{1}{p(s=1 \mid \boldsymbol{x})} - 1\right] \qquad \text{rearrange} \quad (8.13)$$

We can estimate P(s=1|x) using a binary classifier!

Clarifications

- Note Z^{new} is the same for all the data points (i.e., $\forall(\boldsymbol{x}, y)$). It is also a normalization to make sure $\sum_{(\boldsymbol{x},y)} \mathcal{D}^{new}(x, y) = 1$.

$$Z^{new} = \sum_{(\boldsymbol{x},y)} [\mathcal{D}^{base}(x, y) p(s = 0|x)]$$

- Therefore, Z^{new} is not a function of $(\boldsymbol{x}, y)$, in fact it is a constant.
- Similarly for Z^{old}.
- All examples are drawn from fixed base distribution, some are selected to go into D^{new}, some are selected to go into D^{old} according to a selection variable s.

Algorithm 23 SELECTIONADAPTATION($\langle(x_n, y_n)\rangle_{n=1}^{N}$, $\langle z_m\rangle_{m=1}^{M}$, $\mathcal{A}$)

1: $D^{dist} \leftarrow \langle(x_n, +1)\rangle_{n=1}^{N} \cup \langle(z_m, -1)\rangle_{m=1}^{M}$ // assemble data for distinguishing
// between old and new distributions

2: $\hat{p} \leftarrow$ train logistic regression on D^{dist}

3: $D^{weighted} \leftarrow \left\langle\left(x_n, y_n, \frac{1}{\hat{p}(x_n)} - 1\right)\right\rangle_{n=1}^{N}$ // assemble weight classification
// data using selector

4: **return** $\mathcal{A}(D^{weighted})$ // train classifier

We will revisit logistic regression!

Supervised adaptation

Goal: learn a classifier f that achieves low expected loss under new distribution $\mathcal{D}^{new}$

Given labeled training data from old distribution

$$\mathcal{D}^{\text{old}} \quad \langle \boldsymbol{x}_n^{\text{(old)}}, y_n^{\text{(old)}} \rangle_{n=1}^{N}$$

And labeled examples from new distribution

$$\mathcal{D}^{\text{new}} : \langle \boldsymbol{x}_m^{\text{(new)}}, y_m^{\text{(new)}} \rangle_{m=1}^{M}$$

One solution: feature augmentation

Map inputs to a new augmented representation

$$\begin{array}{rcl} & & \quad\text{shared} \qquad\quad \text{old-only} \qquad\qquad \text{new-only} \\ \boldsymbol{x}_n^{(\text{old})} & \mapsto & \left\langle \quad \boldsymbol{x}_n^{(\text{old})} \quad , \; \boldsymbol{x}_n^{(\text{old})} \qquad\qquad , \; \underbrace{0, 0, \dots, 0}_{D\text{-many}} \right\rangle \\ \boldsymbol{x}_m^{(\text{new})} & \mapsto & \left\langle \quad \boldsymbol{x}_m^{(\text{new})} \quad , \; \underbrace{0, 0, \dots, 0}_{D\text{-many}} \quad , \; \boldsymbol{x}_m^{(\text{new})} \qquad\qquad \right\rangle \end{array}$$

One solution: feature augmentation

- Transform D_{old} and D_{new} training examples
- Train a classifier on new representations
- Done!

One solution: feature augmentation

- Adding instance weighting might be useful if N >> M

- Most effective when distributions are "not too close but not too far"
  - In practice, always try "old only", "new only", "union of old and new" as well!

Theorem 9 (Unsupervised Adaptation Bound). *Given a fixed representation and a fixed hypothesis space $\mathcal{F}$, let $f \in \mathcal{F}$ and let $\epsilon^{(\text{best})} = \min_{f^* \in \mathcal{F}} \frac{1}{2}\left[\epsilon^{(\text{old})}(f^*) + \epsilon^{(\text{new})}(f^*)\right]$, then, for all $f \in \mathcal{F}$:*

$$\underbrace{\epsilon^{(\text{new})}(f)}_{\text{error on } \mathcal{D}^{new}} \leq \underbrace{\epsilon^{(\text{old})}(f)}_{\text{error on } \mathcal{D}^{old}} + \underbrace{\epsilon^{(\text{best})}}_{\text{minimal avg error}} + \underbrace{d_{\mathcal{A}}(\mathcal{D}^{old}, \mathcal{D}^{new})}_{\text{distance}} \tag{8.27}$$

Bias is pervasive

- Bias in the labeling
- Sample selection bias
- Bias in choice of labels
- Bias in features or model structure
- Bias in loss function
- Deployed systems create feedback loops

Bias and how to deal with it

- Train/test mismatch
- Unsupervised adaptation
- Supervised adaptation

ACM Code of Ethics

"To minimize the possibility of indirectly harming others, computing professionals must minimize malfunctions by following generally accepted standards for system design and testing. Furthermore, it is often necessary to assess the social consequences of systems to project the likelihood of any serious harm to others. If system features are misrepresented to users, coworkers, or supervisors, the individual computing professional is responsible for any resulting injury."

https://www.acm.org/about-acm/acm-code-of-ethics-and-professional-conduct

UNIVERSITY OF MARYLAND

Furong Huang
4204 IRB, College Park, MD 20740
301.405.8010 / furongh@umd.edu