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: \(\alpha\)-Weighted Binary Classification**

Given:

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

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

**Subsampling Optimality Theorem:**
If the binary classifier achieves a binary error rate of \(\epsilon\), then the error rate of the \(\alpha\)-weighted classifier is \(\alpha \cdot \epsilon\)
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: $\mathbb{E}_{(x,y) \sim \mathcal{D}} [f(x) \neq y]$
**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: $\mathbb{E}_{(x,y) \sim \mathcal{D}}[f(x) \neq y]$
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 OneVersusAllTrain($D^{multiclass}$, BinaryTrain)**

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

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

1. score $\leftarrow \langle 0, 0, \ldots, 0 \rangle$ \hspace{1cm} // initialize K-many scores to zero
2. for $i = 1$ to $K$ do
3. \[ y \leftarrow f_i(\hat{x}) \]
4. \[ \text{score}_i \leftarrow \text{score}_i + y \]
5. end for
6. return $\text{argmax}_k \text{score}_k$
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 $\varepsilon$, then the error rate of the OVA multiclass classifier is at most $(K-1)\varepsilon$

• 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?

\[
\frac{(K - 1)}{K}
\]

Efficiency: \[
\frac{(K - 1)}{K} / 1 = \frac{(K - 1)}{K}
\]
Error bound proof

- 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 = \( \frac{1}{m + 1} \)
  - If there is no false negative: \( \frac{m}{m + 1} \)
    - Efficiency = \( \frac{m}{m + 1} / m = \frac{1}{m + 1} \)
Error bound proof

• What is the worst case scenario?

  – False negative case: efficiency is \( \frac{(K-1)}{K} \)
    • Larger than false positive efficiencies

  – There are \( \mathbf{K} \)-many opportunities to get false negative, overall error bound is \( \mathbf{(K-1)} \varepsilon \)
Reduction 2: AVA

• All versus all (aka all pairs)

• How many binary classifiers does this require?
Algorithm 14 \textbf{AllVersusAllTrain}(D^{\text{multiclass}}, \text{BinaryTrain})

1: \( f_{ij} \leftarrow \emptyset, \forall 1 \leq i < j \leq K \)
2: \textbf{for} \( i = 1 \) \textbf{to} \( K-1 \) \textbf{do}
3: \quad \( D^{\text{pos}} \leftarrow \text{all } x \in D^{\text{multiclass}} \text{ labeled } i \)
4: \quad \textbf{for} \( j = i+1 \) \textbf{to} \( K \) \textbf{do}
5: \quad \quad \( D^{\text{neg}} \leftarrow \text{all } x \in D^{\text{multiclass}} \text{ labeled } j \)
6: \quad \quad \( D^{\text{bin}} \leftarrow \{(x, +1) : x \in D^{\text{pos}}\} \cup \{(x, -1) : x \in D^{\text{neg}}\} \)
7: \quad \quad \( f_{ij} \leftarrow \text{BinaryTrain}(D^{\text{bin}}) \)
8: \quad \textbf{end for}
9: \textbf{end for}
10: \textbf{return} all \( f_{ij} \)s

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

1: \( \text{score} \leftarrow \langle 0, 0, \ldots, 0 \rangle \) \hspace{1cm} // initialize \( K \)-many scores to zero
2: \textbf{for} \( i = 1 \) \textbf{to} \( K-1 \) \textbf{do}
3: \quad \textbf{for} \( j = i+1 \) \textbf{to} \( K \) \textbf{do}
4: \quad \quad \( y \leftarrow f_{ij}(\hat{x}) \)
5: \quad \quad \( \text{score}_i \leftarrow \text{score}_i + y \)
6: \quad \quad \( \text{score}_j \leftarrow \text{score}_j - y \)
7: \quad \textbf{end for}
8: \textbf{end for}
9: \textbf{return} \( \text{argmax}_k \text{score}_k \)
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 \( \varepsilon \), then the error rate of the AVA multiclass classifier is at most \( 2(K-1) \varepsilon \)

• **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!
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: \( D \leftarrow [\ ] \)
2: \( \text{for } n = 1 \text{ to } N \text{ do} \)
3: \( \quad \text{for all } i, j = 1 \text{ to } M \text{ and } i \neq j \text{ do} \)
4: \( \quad \quad \text{if } i \text{ is preferred to } j \text{ on query } n \text{ then} \)
5: \( \quad \quad \quad D \leftarrow D \oplus (x_{nij}, +1) \)
6: \( \quad \quad \text{else if } j \text{ is preferred to } i \text{ on query } n \text{ then} \)
7: \( \quad \quad \quad D \leftarrow D \oplus (x_{nij}, -1) \)
8: \( \quad \text{end if} \)
9: \( \quad \text{end for} \)
10: \( \text{end for} \)
11: \( \text{return } \text{BinaryTrain}(D) \)

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

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

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

Given:

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

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

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

where $\hat{\sigma} = f(x)$
Example of cost functions

\[ \omega(i, j) = \begin{cases} 
1 & \text{if } \min\{i, j\} \leq K \text{ and } i \neq j \\
0 & \text{otherwise}
\end{cases} \]
Algorithm 18 \textsc{RankTrain}(\textbf{D}^{\text{rank}}, \omega, \textbf{BinaryTrain})

\begin{enumerate}
\item $D^{bin} \leftarrow [\ ]$
\item \textbf{for all} $(x, \sigma) \in D^{\text{rank}}$ \textbf{do}
\item \hspace{1em} \textbf{for all} $u \neq v$ \textbf{do}
\item \hspace{2em} $y \leftarrow \text{SIGN} (\sigma_v - \sigma_u)$ \hfill // $y$ is +1 if $u$ is preferred to $v$
\item \hspace{2em} $w \leftarrow \omega (\sigma_u, \sigma_v)$ \hfill // $w$ is the cost of misclassification
\item \hspace{2em} $D^{bin} \leftarrow D^{bin} \oplus (y, w, x_{uv})$
\item \hspace{1em} \textbf{end for}
\item \textbf{end for}
\item \textbf{return} \textbf{BinaryTrain}(\textbf{D}^{bin})
\end{enumerate}
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!
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 $\text{NaiveRankTrain}(\text{RankingData}, \text{BinaryTrain})$

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

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

1: $score \leftarrow \langle 0, 0, \ldots, 0 \rangle$ \hspace{1cm} // initialize $M$-many scores to zero
2: for all $i, j = 1$ to $M$ and $i \neq j$ do
3:     $y \leftarrow f(\hat{x}_{ij})$ \hspace{1cm} // 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 $\text{argsort}(score)$ \hspace{1cm} // return queries sorted by score

Features associated with comparing document $j$ and document $j$ for query $n$.
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 $f : \mathcal{X} \rightarrow \Sigma_M$ minimizing:

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

where $\hat{\sigma} = f(x)$
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 $\text{RankTrain}(D^{rank}, \omega, \text{BinaryTrain})$

1: $D^{bin} \leftarrow [\ ]$
2: for all $(x, \sigma) \in D^{rank}$ do
3:     for all $u \neq v$ do
4:         $y \leftarrow \text{SIGN}(\sigma_v - \sigma_u)$ // $y$ is +1 if $u$ is preferred to $v$
5:         $w \leftarrow \omega(\sigma_u, \sigma_v)$ // $w$ is the cost of misclassification
6:         $D^{bin} \leftarrow D^{bin} \oplus (y, w, x_{uv})$
7:     end for
8: end for
9: return $\text{BinaryTrain}(D^{bin})$
Algorithm 19 RankTest($f$, $\hat{x}$, obj)

1: if obj contains 0 or 1 elements then
2:   return obj
3: else
4:   $p \leftarrow$ randomly chosen object in obj // pick pivot
5:   left $\leftarrow$ [ ] // elements that seem smaller than $p$
6:   right $\leftarrow$ [ ] // elements that seem larger than $p$
7: for all $u \in$ obj \ {$p$} do
8:   $\hat{y} \leftarrow f(x_{up})$ // what is the probability that $u$ precedes $p$
9:   if uniform random variable $< \hat{y}$ then
10:      left $\leftarrow$ left $\oplus$ u
11:   else
12:      right $\leftarrow$ right $\oplus$ u
13:   end if
14: end for
15: left $\leftarrow$ RankTest($f$, $\hat{x}$, left) // sort earlier elements
16: right $\leftarrow$ RankTest($f$, $\hat{x}$, right) // sort later elements
17: return left $\oplus$ $\langle p \rangle$ $\oplus$ 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
  – $\omega$ – ranking