Imbalanced Data and Reductions

CMSC 422
MARINE CARPUAT
marine@cs.umd.edu
Typical Design Process for an ML Application

1. Real world goal: increase revenue
2. R.W. mechanism: better ad display
3. Learning problem: classify click-through
4. Data collect mech: interact w/ cur system
5. Collected data: query, ad, click
6. Data representation: bow^2, +/- click
7. Hypoth. class/ind. bias: dec. tree depth 20
8. Training data selection: subset from apr '16
9. Model training (+hps): final decision tree
10. Predict on test data: subset from may '16
11. Evaluate error: AUC for +/- click predict'n

Deploy!
Imbalanced data distributions

- Sometimes training examples are drawn from an imbalanced distribution

- This results in an imbalanced training set
  - “needle in a haystack” problems
  - E.g., find fraudulent transactions in credit card histories

- Why is this a big problem for the ML algorithms we know?
Recall: Machine Learning as Function Approximation

Problem setting
• Set of possible instances $X$
• Unknown target function $f: X \rightarrow Y$
• Set of function hypotheses $H = \{h | h: X \rightarrow Y\}$

Input
• Training examples $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(N)}, y^{(N)})\}$ of unknown target function $f$

Output
• Hypothesis $h \in H$ that best approximates target function $f$
Recall: Loss Function

\[ l(y, f(x)) \] where \( y \) is the truth and \( f(x) \) is the system’s prediction

\[
e.g. \quad l(y, f(x)) = \begin{cases} 
0 & \text{if } y = f(x) \\
1 & \text{otherwise}
\end{cases}
\]

Captures our notion of what is important to learn
Recall: Expected loss

• $f$ should make good predictions
  – as measured by loss $l$
  – on **future** examples that are also drawn from $D$

• Formally
  – $\varepsilon$, the expected loss of $f$ over $D$ with respect to $l$ should be small

\[
\varepsilon \triangleq \mathbb{E}_{(x,y) \sim D}\{l(y, f(x))\} = \sum_{(x,y)} D(x, y) l(y, f(x))
\]
**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: $\mathbb{E}_{(x,y) \sim \mathcal{D}}[f(x) \neq y]$
We define cost of misprediction as:
\[\alpha \begin{cases} > 1 & \text{for } y = +1 \\ 1 & \text{for } y = -1 \end{cases}\]

Given a good algorithm for solving the binary classification problem, how can I solve the \(\alpha\)-weighted binary classification problem?
Solution: Train a binary classifier on an induced distribution

\[ \text{Algorithm 11 } \text{SubsampleMap}(D_{\text{weighted}}, \alpha) \]

1: while true do
2: \((x, y) \sim D_{\text{weighted}}\) \quad // draw an example from the weighted distribution
3: \(u \sim \text{uniform random variable in } [0, 1]\)
4: if \(y = +1 \or u < \frac{1}{\alpha}\) then
5: \quad return \((x, y)\)
6: end if
7: end while
Subsampling optimality

• **Theorem:** If the binary classifier achieves a binary error rate of $\varepsilon$, then the error rate of the $\alpha$-weighted classifier is $\alpha \varepsilon$

• Let’s prove it.
  (see also CIML 6.1)
Strategies for inducing a new binary distribution

• Undersample the negative class

• Oversample the positive class
Strategies for inducing a new binary distribution

- Undersample the negative class
  - More computationally efficient

- Oversample the positive class
  - Base binary classifier might do better with more training examples
  - Efficient implementations incorporate weight in algorithm, instead of explicitly duplicating data!
Algorithm 1 \textbf{DecisionTreeTrain}(data, remaining features)

1: \texttt{guess} $\leftarrow$ most frequent answer in data  \hspace{1em} // default answer for this data
2: \textbf{if} the labels in data are unambiguous \textbf{then}
3: \hspace{1em} \textbf{return} \textbf{Leaf}(\texttt{guess}) \hspace{1em} // base case: no need to split further
4: \hspace{1em} \textbf{else if} remaining features is empty \textbf{then}
5: \hspace{2em} \textbf{return} \textbf{Leaf}(\texttt{guess}) \hspace{1em} // base case: cannot split further
6: \hspace{1em} \textbf{else}
7: \hspace{2em} \textbf{for all} $f \in$ remaining features \textbf{do}
8: \hspace{3em} \texttt{NO} $\leftarrow$ the subset of data on which $f=\text{no}$
9: \hspace{3em} \texttt{YES} $\leftarrow$ the subset of data on which $f=\text{yes}$
10: \hspace{3em} score[$f$] $\leftarrow$ \# of majority vote answers in \texttt{NO}
11: \hspace{3em} + \# of majority vote answers in \texttt{YES}
12: \hspace{3em} \hspace{1em} // the accuracy we would get if we only queried on $f$
13: \textbf{end for}
14: \hspace{1em} $f \leftarrow$ the feature with maximal score($f$)
15: \hspace{1em} \texttt{NO} $\leftarrow$ the subset of data on which $f=\text{no}$
16: \hspace{1em} \texttt{YES} $\leftarrow$ the subset of data on which $f=\text{yes}$
17: \hspace{1em} \texttt{left} $\leftarrow$ \textbf{DecisionTreeTrain}(\texttt{NO}, remaining features \setminus \{f\})
18: \hspace{1em} \texttt{right} $\leftarrow$ \textbf{DecisionTreeTrain}(\texttt{YES}, remaining features \setminus \{f\})
19: \hspace{1em} \textbf{return} \textbf{Node}($f$, \texttt{left}, \texttt{right})
20: \textbf{end if}
Reductions

• Idea is to re-use simple and efficient algorithms for binary classification to perform more complex tasks

• Works great in practice:
  – E.g., Vowpal Wabbit
Learning with Imbalanced Data is an Example of Reduction

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

**Subsampling Optimality Theorem:**
If the binary classifier achieves a binary error rate of \( \varepsilon \), then the error rate of the \( \alpha \)-weighted classifier is \( \alpha \varepsilon \)
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?
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
What you should know

• How can we take the standard binary classifier and adapt it to handle problems with
  – Imbalanced data distributions
  – Multiclass classification problems

• Algorithms & guarantees on error rate

• Fundamental ML concept: reduction
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: $D^{bin} \leftarrow$ relabel $D_{multiclass}$ so class $i$ is positive and $\neg i$ is negative
3: $f_i \leftarrow$ 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: score$_i \leftarrow$ score$_i + y$
5: end for
6: 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 $\varepsilon$, 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: \[
\frac{(K - 1)}{K} \times \frac{1}{1} = \frac{(K - 1)}{K}
\]
Error bound proof

• If we have k **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 = $1 / (k + 1)$
  – Otherwise $k / (k + 1)$
    • Efficiency = $k / (k + 1) / k = 1 / (k + 1)$
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** \((K-1) \varepsilon\)
Reduction 2: AVA

• All versus all (aka all pairs)

• How many binary classifiers does this require?
Algorithm 14 **AllVersusAllTrain**($D_{multiclass}$, **BinaryTrain**)

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

Algorithm 15 **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: for \( i = 1 \) to \( K-1 \) do
3: \( \text{for } j = i+1 \text{ to } K \text{ do} \)
4: \( y \leftarrow f_{ij}(\hat{x}) \)
5: \( \text{score}_i \leftarrow \text{score}_i + y \)
6: \( \text{score}_j \leftarrow \text{score}_j - y \)
7: \( \text{end for} \)
8: \( \text{end for} \)
9: 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