Imbalanced Data and Reductions

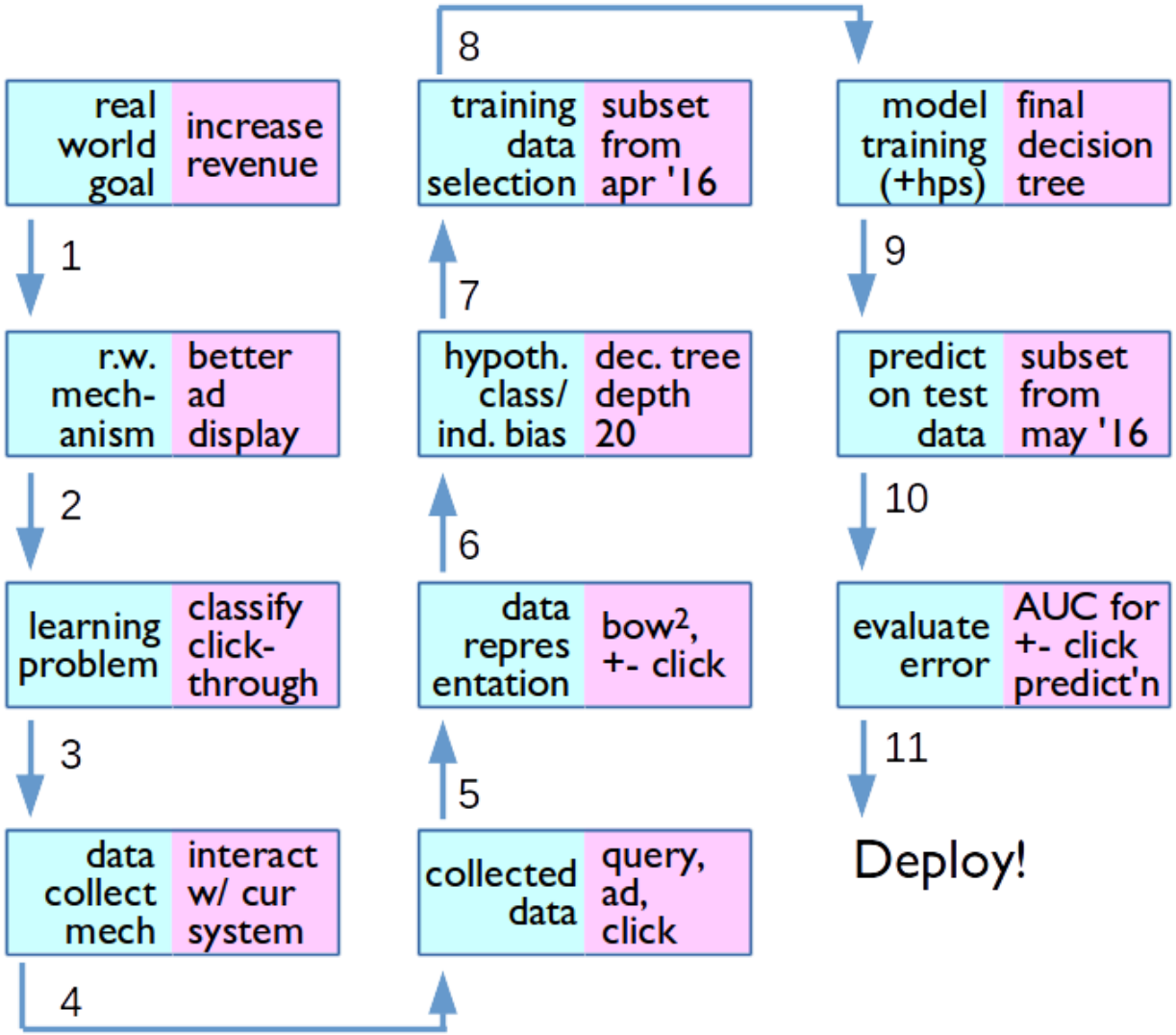
CMSC 422

MARINE CARPUAT
marine@cs.umd.edu

Today's topics

- Using standard binary classifiers to solve other problems
  - Weighted classification
  - Multiclass classification
- Algorithms & guarantees on error rate
- Fundamental ML concept: reduction

Typical Design Process for an ML Application



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 $H = \{h \mid h: X \rightarrow Y\}$

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: Expected loss

- f should make good predictions
  - as measured by loss l
  - on **future** examples that are also drawn from D
- Formally
  - ε , 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: $\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[f(\boldsymbol{x}) \neq y\right]$

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

Given a good algorithm for solving the binary classification problem, how can I solve the α-weighted binary classification problem?

We define cost of misprediction as:

α > 1 for y = +1

1 for y = -1

Solution: Train a binary classifier on an induced distribution

Algorithm 11 SUBSAMPLEMAP($\mathcal{D}^{weighted}$, α)

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

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

- 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 DecisionTreeTrain(*data*, *remaining features*)

```
guess ← most frequent answer in data                    // default answer for this data
if the labels in data are unambiguous then
    return Leaf(guess)                                  // base case: no need to split further
else if remaining features is empty then
    return Leaf(guess)                                  // base case: cannot split further
else                                                    // we need to query more features
    for all f ∈ remaining features do
        NO ← the subset of data on which f=no
        YES ← the subset of data on which f=yes
        score[f] ← # of majority vote answers in NO
                   + # of majority vote answers in YES
                                                        // the accuracy we would get if we only queried on f
    end for
    f ← the feature with maximal score(f)
    NO ← the subset of data on which f=no
    YES ← the subset of data on which f=yes
    left ← DecisionTreeTrain(NO, remaining features \ {f})
    right ← DecisionTreeTrain(YES, remaining features \ {f})
    return Node(f, left, right)
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: 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: $\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\alpha^{y=1}\left[f(x) \neq y\right]\right]$

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?

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

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

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

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

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

1: $score \leftarrow \langle 0, 0, \ldots, 0 \rangle$ // initialize K-many scores to zero
2: **for** $i = 1$ **to** K **do**
3: $\quad y \leftarrow f_i(\hat{\boldsymbol{x}})$
4: $\quad score_i \leftarrow score_i + y$
5: **end for**
6: **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 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?

Reduction 2: AVA

- All versus all (aka all pairs)

- How many binary classifiers does this require?

Algorithm 14 AllVersusAllTrain($\mathbf{D}^{multiclass}$, 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
    $\mathbf{D}^{bin} \leftarrow \{(\boldsymbol{x}, +1) : \boldsymbol{x} \in \mathbf{D}^{\text{pos}}\} \cup \{(\boldsymbol{x}, -1) : \boldsymbol{x} \in \mathbf{D}^{\text{neg}}\}$
    $f_{ij} \leftarrow$ BinaryTrain($\mathbf{D}^{bin}$)
  end for
end for
return all f_{ij}s

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

$score \leftarrow \langle 0, 0, \ldots, 0 \rangle$ // 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?

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