Slides adapted from Prof Carpuat and Duraiswami

# CMSC 422 Introduction to Machine Learning

# Lecture 11 Imbalanced Data and Reductions

Furong Huang / furongh@umd.edu

UNIVERSITY OF MARYLAND

# Formalizing Errors

The learned classifier

$\mathcal{F}$ set of all possible classifiers using a fixed representation

$$\text{error}(f) = \underbrace{\left[\text{error}(f) - \min_{f^* \in \mathcal{F}} \text{error}(f^*)\right]}_{\textbf{estimation error}} + \underbrace{\left[\min_{f^* \in \mathcal{F}} \text{error}(f^*)\right]}_{\textbf{approximation error}}$$

How far is the learned classifier f from the optimal classifier f*?

Quality of the model family
aka hypothesis class

# The bias/variance trade-off

- Trade-off between
  - approximation error (bias)
  - estimation error (variance)

- Example:
  - Consider the always positive classifier
    - Low variance as a function of a random draw of the training set
    - Strongly biased toward predicting +1 no matter what the input

# 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 $\{(x^{(1)}, y^{(1)}), \dots (x^{(N)}, y^{(N)})\}$ 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

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

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

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($\mathcal{D}^{weighted}$, $\alpha$)

```
while true do
    (x, y) ~ D^weighted                 // draw an example from the weighted distribution
    u ~ uniform random variable in [0, 1]
    if y = +1 or u < 1/α then
        return (x, y)
    end if
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

Proof omitted in class.

(see also CIML 6.1)

*Proof of Theorem 3.* Let $\mathcal{D}^w$ be the original distribution and let $\mathcal{D}^b$ be the induced distribution. Let $f$ be the binary classifier trained on data from $\mathcal{D}^b$ that achieves a binary error rate of $\epsilon^b$ on that distribution. We will compute the expected error $\epsilon^w$ of $f$ on the weighted problem:

$$\epsilon^w = \mathbb{E}_{(x,y)\sim\mathcal{D}^w}\left[\alpha^{y=1}\left[f(x) \neq y\right]\right] \tag{6.1}$$

$$= \sum_{x\in\mathcal{X}}\sum_{y\in\pm1} \mathcal{D}^w(x,y)\alpha^{y=1}\left[f(x) \neq y\right] \tag{6.2}$$

$$= \alpha\sum_{x\in\mathcal{X}}\left(\mathcal{D}^w(x,+1)\left[f(x) \neq +1\right] + \mathcal{D}^w(x,-1)\frac{1}{\alpha}\left[f(x) \neq -1\right]\right) \tag{6.3}$$

$$= \alpha\sum_{x\in\mathcal{X}}\left(\mathcal{D}^b(x,+1)\left[f(x) \neq +1\right] + \mathcal{D}^b(x,-1)\left[f(x) \neq -1\right]\right) \tag{6.4}$$

$$= \alpha\mathbb{E}_{(x,y)\sim\mathcal{D}^b}\left[f(x) \neq y\right] \tag{6.5}$$

$$= \alpha\epsilon^b \tag{6.6}$$

And we're done! (We implicitly assumed $\mathcal{X}$ is discrete. In the case of continuous data, you need to replace all the sums over $x$ with integrals over $x$, but the result still holds.) $\square$

# Strategies for inducing a new binary distribution

- Undersample the negative class (dominant class)
- Oversample the positive class (subordinate 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!

# Reductions

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

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

# 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

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

**for** $i = 1$ **to** $K$ **do**
  $\mathbf{D}^{bin} \leftarrow$ relabel $\mathbf{D}^{multiclass}$ so class $i$ is positive and $\neg i$ is negative
  $f_i \leftarrow \textsc{BinaryTrain}(\mathbf{D}^{bin})$
**end for**
**return** $f_1, \dots, f_K$

**Algorithm 13** $\textsc{OneVersusAllTest}(f_1, \dots, f_K, \hat{\boldsymbol{x}})$

$score \leftarrow \langle 0, 0, \dots, 0 \rangle$ // initialize $K$-many scores to zero
**for** $i = 1$ **to** $K$ **do**
  $y \leftarrow f_i(\hat{\boldsymbol{x}})$
  $score_i \leftarrow score_i + y$
**end for**
**return** $\operatorname{argmax}_k score_k$

# 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 ε, then the error rate of the OVA multiclass classifier is at most (K-1) ε

- 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

1. 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?

$$(K - 1) / K$$

Efficiency: $[(K - 1) / K] / 1 = (K - 1) / K$

# Error bound proof

2. 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 =1 / (m + 1)
- Otherwise  m / ( m + 1)
  - Efficiency = [m / (m + 1)] / m = 1 / ( m + 1)

# Error bound proof

## 3. 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) ε**

UNIVERSITY OF
MARYLAND

**Furong Huang**
4124 IRB, College Park, MD 20740
301.405.8010 / furongh@umd.edu