Practical Issues: Evaluation, beyond accuracy

- So far we’ve measured classification performance using **accuracy**

- But this is not a good metric when some errors matter more than others
  - Given medical record, predict whether patient has cancer or not
  - Given a document collection and a query, find documents in collection that are relevant to query
Imagine we are addressing a document retrieval task for a given query, where +1 means that the document is relevant and -1 means that the document is not relevant.

We can categorize predictions as:
- true/false positives
- true/false negatives

<table>
<thead>
<tr>
<th></th>
<th>Gold label = +1</th>
<th>Gold label = -1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prediction = +1</td>
<td>tp</td>
<td>fp</td>
</tr>
<tr>
<td>Prediction = -1</td>
<td>fn</td>
<td>tn</td>
</tr>
</tbody>
</table>
Precision and recall

- **Precision**: % of positive predictions that are correct
  \[ \text{Precision} = \frac{tp}{tp + fp} \]
  Denominator: # of positive predictions

- **Recall**: % of positive gold labels that are found
  \[ \text{Recall} = \frac{tp}{tp + fn} \]
  Denominator: # of positive gold labels

<table>
<thead>
<tr>
<th></th>
<th>Gold label = +1</th>
<th>Gold label = -1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prediction</td>
<td>tp</td>
<td>fp</td>
</tr>
<tr>
<td>Prediction</td>
<td>fn</td>
<td>tn</td>
</tr>
<tr>
<td>Prediction</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
A combined measure: F

- A combined measure that assesses the P/R tradeoff is F measure

\[
P = \frac{tp}{tp + fp} \\
R = \frac{tp}{tp + fn}
\]

\[
F = \frac{1}{\alpha \frac{1}{P} + (1 - \alpha) \frac{1}{R}} = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}
\]

- People usually use balanced F-1 measure
  - i.e., with \( \beta = 1 \) (that is, \( \alpha = \frac{1}{2} \))
  - Harmonic mean \( F = \frac{2PR}{P+R} \)
Formalizing Errors

The learned classifier $f$ is a classifier using a fixed representation from the set of all possible classifiers $\mathcal{F}$. How far is the learned classifier $f$ from the optimal classifier $f^*$?

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

- Estimation error
- Approximation error

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
Recap: practical issues

- Learning algorithm is only one of many steps in designing a ML application

- Many things can go wrong, but there are practical strategies for
  - Improving inputs
  - Evaluating
  - Tuning
  - Debugging

- Fundamental ML concepts: estimation vs. approximation error
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: Loss Function

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

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

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
  - $\mathcal{E}$, the expected loss of $f$ over $D$ with respect to $l$ should be small
  - $\mathcal{E} \triangleq \mathbb{E}_{(x, y) \sim D} \{ l(y, f(x)) \} = \sum_{(x, y)} D(x, y)l(y, f(x))$
**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:

\[ \alpha > 1 \text{ for } y = +1 \]
\[ 1 \text{ for } y = -1 \]

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

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

1: \textbf{while} true \textbf{do}
2: \hspace{1em} (x, y) \sim D_{\text{weighted}} \quad \text{// draw an example from the weighted distribution}
3: \hspace{1em} u \sim \text{uniform random variable in } [0, 1]
4: \hspace{1em} \text{if } y = +1 \text{ or } u < \frac{1}{\alpha} \text{ then}
5: \hspace{2em} \text{return } (x, y)
6: \hspace{1em} \text{end if}
7: \textbf{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)
Proof of Theorem 3. Let $D^w$ be the original distribution and let $D^b$ be the induced distribution. Let $f$ be the binary classifier trained on data from $D^b$ 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:

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

$$
= \sum_{x \in \mathcal{X}} \sum_{y \in \{-1, 1\}} D^w(x, y) \alpha^y=1 [f(x) \neq y]
$$

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

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

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

$$
= \alpha \epsilon^b
$$

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.)
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

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: 

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

**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 \textbf{OneVersusAllTrain}(D^{\text{multiclass}}, \text{BinaryTrain})

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

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

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

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

Efficiency: \[
\frac{\left( \frac{K - 1}{K} \right)}{1} = \frac{K - 1}{K}
\]
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

\[
\text{Efficiency} = \frac{1}{m + 1}
\]

Otherwise \( m \) \( \text{false positives} \):

\[
\text{Efficiency} = \frac{m}{m + 1}
\]

\[
\text{Efficiency} = \left[ \frac{m}{m + 1} \right] / m = \frac{1}{m + 1}
\]
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** \((K-1) \varepsilon\)
Reduction 2: AVA

All versus all (aka all pairs)

How many binary classifiers does this require?
Algorithm 14 \textbf{AllVersusAllTrain}(D^{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: \hspace{1em} \( D^{pos} \leftarrow \) all \( x \in D^{multiclass} \) labeled \( i \)
4: \hspace{1em} \textbf{for} \( j = i+1 \) \textbf{to} \( K \) \textbf{do}
5: \hspace{2em} \( D^{neg} \leftarrow \) all \( x \in D^{multiclass} \) labeled \( j \)
6: \hspace{2em} \( D^{bin} \leftarrow \{(x, +1) : x \in D^{pos}\} \cup \{(x, -1) : x \in D^{neg}\} \)
7: \hspace{1em} \( f_{ij} \leftarrow \text{BinaryTrain}(D^{bin}) \)
8: \hspace{1em} \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{1em} // initialize \( K \)-many scores to zero
2: \textbf{for} \( i = 1 \) \textbf{to} \( K-1 \) \textbf{do}
3: \hspace{1em} \textbf{for} \( j = i+1 \) \textbf{to} \( K \) \textbf{do}
4: \hspace{2em} \( y \leftarrow f_{ij}(\hat{x}) \)
5: \hspace{2em} \( \text{score}_i \leftarrow \text{score}_i + y \)
6: \hspace{2em} \( \text{score}_j \leftarrow \text{score}_j - y \)
7: \hspace{1em} \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
Furong Huang
3251 A.V. Williams, College Park, MD 20740
301.405.8010 / furongh@cs.umd.edu