Learning reductions

1. Problem: importance-weighted classification
   - Reduction: rejection sampling
     (Reduces problem to unweighted classification.)
2. Problem: multi-class classification
   - Reduction #1: One-Against-All
   - Reduction #2: Error Correcting Output Codes
     (Both reduce problem to binary classification.)

In machine learning, typically have
- Problem 1: the problem you have to solve for a real application
- Problem 2: a well-studied problem in machine learning
- Problem instance: training data and (implicitly) a probability distribution $P$
- Solution: prediction functions
- $A$: the latest, greatest learning algorithm for Problem 2

Examples
1. Problem: importance-weighted classification
   - Reduction: rejection sampling
     (Reduces problem to unweighted classification.)
2. Problem: multi-class classification
   - Reduction #1: One-Against-All
   - Reduction #2: Error Correcting Output Codes
     (Both reduce problem to binary classification.)

Problem:
- Setting: Random triple $(X, Y, C) \sim P$ for some probability distribution $P$ over $\mathcal{X} \times \mathcal{Y} \times \mathbb{R}_+$. $C =$ importance weight for labeled example $(X, Y)$.
- Goal: Function $f : \mathcal{X} \rightarrow \mathcal{Y}$ with small importance-weighted error:
  $$\mathbb{E}[C \cdot 1\{f(X) \neq Y\}].$$

Problem instance:
- Training data $S$: collection of triples $(x, y, c) \in \mathcal{X} \times \mathcal{Y} \times \mathbb{R}_+$, presumed to be drawn i.i.d. from $P$.

Where it comes up:
- Class-specific weights: e.g., $C = 100 \iff Y = 0$ (and $C = 1$ otherwise).
- Input-specific weights: e.g., $C = 100 \iff X \in \mathcal{X}_0$ (and $C = 1$ o.w.).
- Boosting, domain adaptation, causal inference, . . .

(Note: many learning algorithms natively handle importance weights.)

Would like to reduce to (unweighted) classification.
**The rejection sampling reduction**

**Main idea:** Transform training data $S$ so it looks like it came from a distribution $P'$, where

$$
E_{(X,Y,C) \sim P} \left[ C \cdot 1 \{ f(X) \neq Y \} \right] = E_{(X',Y') \sim P'} \left[ 1 \{ f(X') \neq Y' \} \right].
$$

**Instance mapping procedure**

**Input** Training data $S$ from $\mathcal{X} \times \mathcal{Y} \times \mathbb{R}^+$.

1: Initialize $S' = \emptyset$.
2: Let $c_{\text{max}} := \max(x,y,c) \in S$.
3: for each $(x,y,c) \in S$ do
4:   Toss a coin with heads bias $\frac{c}{c_{\text{max}}}$.
5:   If heads, keep example—put $(x,y)$ into $S'$.
6:   If tails, discard example.
7: end for
8: return Training data $S'$ from $\mathcal{X} \times \mathcal{Y}$.

**Solution mapping procedure:** identity map

**Why rejection sampling works:** (Assume for simplicity that $c_{\text{max}} = 1$.)

Define random variable $Q := 1 \{ \text{Keep example } (X,Y) \}$

which, after conditioning on $(X,Y,C)$, has mean $C$.

Distribution of examples in $S'$ is same as that of $(X,Y)$ conditioned on $Q = 1$.

Moreover,

$$
E \left[ Q \cdot 1 \{ f(X) \neq Y \} \right] = E \left[ E \left[ Q \cdot 1 \{ f(X) \neq Y \} \mid (X,Y,C) \right] \right] = E \left[ C \cdot 1 \{ f(X) \neq Y \} \right]
$$

**Conclusion:**

Prediction error w.r.t. $P'$ $\propto$ importance-weighted error w.r.t. $P$.

---

**Multi-class classification**

**Problem:**

- **Setting:** Random pair $(X,Y) \sim P$ for some probability distribution $P$ over $\mathcal{X} \times \{1,2,\ldots,K\}$.
- **Goal:** Function $f: \mathcal{X} \rightarrow \mathcal{Y}$ with small prediction error $P(f(X) \neq Y)$.

**Problem instance:**

- Training data $S$: collection of pairs $(x,y) \in \mathcal{X} \times \{1,2,\ldots,K\}$, presumed to be drawn i.i.d. from $P$.

Would like to reduce to binary classification.

---

**One-Against-All reduction**

**Main idea:** Create $K$ binary classification problems

$\text{given } x \in \mathcal{X}, \text{ predict whether or not } y = i.$

Create $K$ examples from each $(x,y) \in S$:

$$(x,y) \rightarrow \begin{cases} 
(x, 1\{y = 1\}) & \rightarrow \ S'_1 \\
(x, 1\{y = 2\}) & \rightarrow \ S'_2 \\
\vdots & \vdots \\
(x, 1\{y = K\}) & \rightarrow \ S'_K
\end{cases}$$

---

**The rejection sampling reduction**

**Conclusion:**

Prediction error w.r.t. $P'$ $\propto$ importance-weighted error w.r.t. $P$. 
**One-Against-All reduction**

**Instance mapping procedure**

- **Input** Training data $S$ from $\mathcal{X} \times \{1, 2, \ldots, K\}$.
  1. Initialize empty sets $S'_1, S'_2, \ldots, S'_K$.
  2. for each $(x, y) \in S$ do
  3. for each $i = 1, 2, \ldots, K$ do
  4. \hspace{1em} Put $(x, 1 \{y = i\}) \in \mathcal{X} \times \{0, 1\}$ into $S'_i$.
  5. end for
  6. end for
  7. return Training data sets $S'_1, S'_2, \ldots, S'_K$ from $\mathcal{X} \times \{0, 1\}$.

**Solution mapping procedure**

- Input $K$ binary predictors $f'_1, f'_2, \ldots, f'_K : \mathcal{X} \to \{0, 1\}$.
- return Function $f : \mathcal{X} \to \{1, 2, \ldots, K\}$ where
  \[
  f(x) = \arg \max_{i \in \{1, 2, \ldots, K\}} f'_i(x) \quad \text{(breaking ties arbitrarily)}.
  \]

  This should seem weird!

---

**Problem with OAA**

**OAA multi-class predictor:**

\[
 f(x) = \arg \max_{i \in \{1, 2, \ldots, K\}} f'_i(x) .
\]

Only get correct classification on $(x, y)$ if $f'_y(x) = 1$ and $f'_i(x) = 0$ for all $i \neq y$.
(Could err if any of the $f'_i$ errs!)

**Solution:** use conditional probability estimation

\[
 f'_i(x) = \text{estimate of } P(Y = i \mid X = x) .
\]

---

**Error correcting output codes**

**Main idea:** Create $m$ binary classification problems given $x \in \mathcal{X}$, predict whether or not $y \in L_i$ for $m$ pre-specified subsets $L_i \subset \{1, 2, \ldots, K\}$.

Subsets specified using **error correcting codes** ($m \times K$ binary matrix):

\[
\begin{bmatrix}
\vdots \\
\mathbf{m} \\
\vdots
\end{bmatrix}
\begin{bmatrix}
\vdots \\
\mathbf{K} \\
\vdots
\end{bmatrix}
\]

Let $i$-th column $c_i \in \{0, 1\}^m$ be the **code word** for class $i \in \{1, 2, \ldots, K\}$.

Use $m$ binary classifiers $f'_1, f'_2, \ldots, f'_m$ to predict entire code word.

**Error correction:** can still get correct multi-class prediction even if several binary classifiers err.

Still also useful to use conditional probability estimation

\[
 f'_i(x) = \text{estimate of } P(Y \in L_i \mid X = x)
\]

("Probabilistic ECOC").

**Caveat:** binary prediction problems could be challenging / unnatural (e.g., predict if handwritten digit is an even digit or not).
Comparing OAA and ECOC

Empirical comparison

- Eight multi-class problems (from the UCI repository).
- $A = \text{classregtree}$ from the MATLAB statistics toolbox, estimate conditional probabilities using square loss.
- ECOC based on Hadamard matrix (similar to Fourier transform).

<table>
<thead>
<tr>
<th>Data set</th>
<th>Number of classes</th>
<th>One-against-all</th>
<th>ECOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>ecoli</td>
<td>8</td>
<td>0.0985</td>
<td>0.0517</td>
</tr>
<tr>
<td>glass</td>
<td>6</td>
<td>0.3874</td>
<td>0.3462</td>
</tr>
<tr>
<td>pendigits</td>
<td>10</td>
<td>0.0985</td>
<td>0.0517</td>
</tr>
<tr>
<td>satimage</td>
<td>6</td>
<td>0.1679</td>
<td>0.1376</td>
</tr>
<tr>
<td>soybean</td>
<td>19</td>
<td>0.6580</td>
<td>0.5993</td>
</tr>
<tr>
<td>splice</td>
<td>3</td>
<td><strong>0.0642</strong></td>
<td>0.0699</td>
</tr>
<tr>
<td>vowel</td>
<td>11</td>
<td>0.6356</td>
<td>0.5780</td>
</tr>
<tr>
<td>yeast</td>
<td>10</td>
<td>0.4893</td>
<td>0.4479</td>
</tr>
</tbody>
</table>

Summary

- **Reductions**: reuse existing technology to solve new problems.
  - Multi-class (OAA, ECOC, tournaments, . . .)
  - Multi-label prediction
  - Ranking
  - Sequence prediction
  - . . .
- **Lots of different problems and objectives** beyond binary classification and prediction error—can be application-/domain-specific.
- **Very useful tools**:
  - Importance-weighted classification (using reductions)
  - Conditional probability estimation (using certain loss functions)