COMS 4771 Lecture 13

1. Beyond prediction error
2. Beyond binary classification

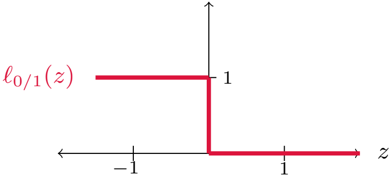
Beyond prediction error

Prediction error / zero-one loss

P is a distribution over $\mathcal{X} \times \{-1, +1\}$, and $(X, Y) \sim P$.

For any classifier $f \colon \mathcal{X} \to \{-1, +1\}$,

$$\mathrm{err}(f) \;=\; \Pr\Big[f(X) \neq Y\Big] \;=\; \mathbb{E}\Big[\ell_{0/1}(Y f(X))\Big].$$

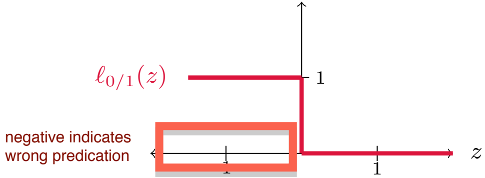


Prediction error / zero-one loss

P is a distribution over $\mathcal{X} \times \{-1,+1\}$, and $(X,Y) \sim P$.

For any classifier $f \colon \mathcal{X} \to \{-1,+1\}$,

$$\operatorname{err}(f) \;=\; \Pr\Big[f(X) \neq Y\Big] \;=\; \mathbb{E}\Big[\ell_{0/1}(Y f(X))\Big].$$



Also works with **real-valued predictors** $f \colon \mathcal{X} \to \mathbb{R}$; for example:

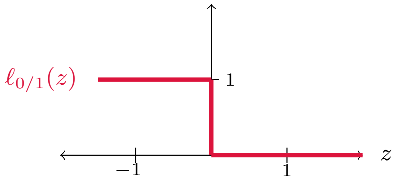
- **Linear classifiers**: $\boldsymbol{x} \mapsto \langle \boldsymbol{w}, \boldsymbol{x} \rangle - \theta$.
- **(Binary) plug-in classifiers** $x \mapsto \widehat{\Pr}[Y = +1 \mid X = x] - 1/2$.

Prediction error / zero-one loss

P is a distribution over $\mathcal{X} \times \{-1, +1\}$, and $(X, Y) \sim P$.

For any classifier $f \colon \mathcal{X} \to \{-1, +1\}$,

$$\operatorname{err}(f) = \Pr\Big[f(X) \neq Y\Big] = \mathbb{E}\Big[\ell_{0/1}(Y f(X))\Big].$$



Also works with **real-valued predictors** $f \colon \mathcal{X} \to \mathbb{R}$; for example:

- **Linear classifiers**: $\boldsymbol{x} \mapsto \langle \boldsymbol{w}, \boldsymbol{x} \rangle - \theta$.
- **(Binary) plug-in classifiers**: $x \mapsto \widehat{\Pr}[Y = +1 \mid X = x] - 1/2$.

Often useful to adjust threshold (e.g., θ and $1/2$ above).

adjust the \theta to
minimize the lose function

Uses for adjusting threshold t

Often have **different costs for different kinds of mistakes** (recall HW1):

	$f(X) \le t$	$f(X) > t$
$Y = -1$	0	c
$Y = +1$	$1 - c$	0

Uses for adjusting threshold t

Often have **different costs for different kinds of mistakes** (recall HW1):

	$f(X) \leq t$	$f(X) > t$
$Y = -1$	0	c
$Y = +1$	$1 - c$	0

Also, often interested in **different performance criteria**.

- **Precision**:

$$\Pr[Y = +1 \mid f(X) > t]$$

- **Recall** (a.k.a. **Sensitivity**, **True Positive Rate**):

$$\Pr[f(X) > t \mid Y = +1]$$

- **Specificity**:

$$\Pr[f(X) \leq t \mid Y = -1]$$

- **False Positive Rate**:

$$\Pr[f(X) > t \mid Y = -1]$$

Conditional probability estimation

Sometimes would like real-valued predictor f to be related to the **conditional probability function η**:

$$\eta(x) := \Pr[Y = +1 \mid X = x].$$

Conditional probability estimation

Sometimes would like real-valued predictor f to be related to the **conditional probability function** η:

$$\eta(x) := \Pr[Y = +1 \mid X = x].$$

Option #1: Can use **generative models** to construct **plug-in classifier**, but could fail if class conditional distributions are not accurate.

Conditional probability estimation

motivation

Sometimes would like real-valued predictor f to be related to the **conditional probability function** η:

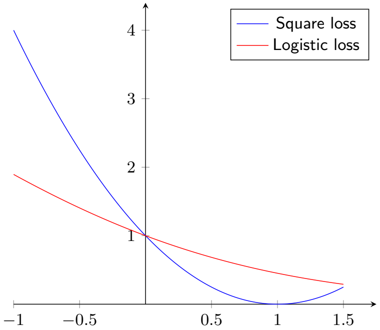
$$\eta(x) := \Pr[Y = +1 \mid X = x].$$

for a random x

Option #1: Can use **generative models** to construct **plug-in classifier**, but could fail if class conditional distributions are not accurate.

Option #2: Can use a loss function that is minimized by η (or some invertible transformation thereof).

Eliciting conditional probabilities



Goal: loss function that is minimized by (some invertible transformation of) the conditional probability function

$$\eta(x) = \Pr[Y = +1 \mid X = x].$$

- **Square loss**: $\ell_{\mathrm{sq}}(z) = (1 - z)^2$

  $\mathbb{E}[\ell_{\mathrm{sq}}(Y f(x))|X = x]$ is minimized by $f(x) = 2\eta(x) - 1$.

- **Logistic loss**: $\ell_{\log}(z) = \ln(1 + \exp(-z))$

  $\mathbb{E}[\ell_{\log}(Y f(x))|X = x]$ is minimized by $f(x) = \ln\left(\frac{\eta(x)}{1-\eta(x)}\right)$.

Using loss functions: easy with linear/affine functions whenever the loss function ℓ is a convex function (due to affine composition rule).

$$\min_{\boldsymbol{w}\in\mathbb{R}^d} \qquad R(\boldsymbol{w}) + \frac{1}{n}\sum_{i=1}^{n} \ell(\boldsymbol{w}; \boldsymbol{x}^{(i)}, y^{(i)})$$

(Here, the regularization function R is also assumed to be convex.)

Eliciting conditional probabilities

Using loss functions: easy with linear/affine functions whenever the loss function ℓ is a convex function (due to affine composition rule).

$$\min_{\boldsymbol{w} \in \mathbb{R}^d} \qquad R(\boldsymbol{w}) + \frac{1}{n} \sum_{i=1}^{n} \ell(\boldsymbol{w}; \boldsymbol{x}^{(i)}, y^{(i)})$$

(Here, the regularization function R is also assumed to be convex.)

Good news: Both ℓ_{sq} and ℓ_{log} are convex!

A non-example:

- **Hinge loss**: $\ell_{\mathrm{hl}}(z) = \max\{0,\, 1 - z\}$

  $\mathbb{E}[\ell_{\mathrm{hl}}(Y f(x)) | X = x]$ is minimized by $f(x) = \mathrm{sign}(2\eta(x) - 1)$.

  Can't recover $\eta(x)$ from $f(x)$.

A non-example:

- **Hinge loss**: $\ell_{\text{hl}}(z) = \max\{0,\, 1 - z\}$

  $\mathbb{E}[\ell_{\text{hl}}(Yf(x))|X = x]$ is minimized by $f(x) = \operatorname{sign}(2\eta(x) - 1)$.

  Can't recover $\eta(x)$ from $f(x)$.

Caveat: Might not be possible to represent

$$\boldsymbol{x} \mapsto 2\eta(\boldsymbol{x}) - 1 \quad \text{or} \quad \boldsymbol{x} \mapsto \ln\left(\frac{\eta(\boldsymbol{x})}{1 - \eta(\boldsymbol{x})}\right)$$

as (say) a linear function $\boldsymbol{x} \mapsto \langle \boldsymbol{w}, \boldsymbol{x} \rangle$.

Eliciting conditional probabilities

A non-example:

- **Hinge loss**: $\ell_{\mathrm{hl}}(z) = \max\{0,\, 1 - z\}$

  $\mathbb{E}[\ell_{\mathrm{hl}}(Yf(x))|X = x]$ is minimized by $f(x) = \mathrm{sign}(2\eta(x) - 1)$.

  Can't recover $\eta(x)$ from $f(x)$.

Caveat: Might not be possible to represent

$$\boldsymbol{x} \mapsto 2\eta(\boldsymbol{x}) - 1 \quad \text{or} \quad \boldsymbol{x} \mapsto \ln\left(\frac{\eta(\boldsymbol{x})}{1 - \eta(\boldsymbol{x})}\right)$$

as (say) a linear function $\boldsymbol{x} \mapsto \langle \boldsymbol{w}, \boldsymbol{x} \rangle$.

Common solutions: enhance the feature space via feature expansion or kernels, or use more flexible models—discussed later.

Beyond binary classification

Reductions

Suppose we have a good algorithm $\mathcal{A}$ for learning binary classifiers.

Question: Can we use $\mathcal{A}$ to solve other machine learning problems?

Reductions

Suppose we have a good algorithm $\mathcal{A}$ for learning binary classifiers.

Question: Can we use $\mathcal{A}$ to solve other machine learning problems?

A **reduction** from Problem 1 to Problem 2 is specified by two procedures:

- Instance map R: transforms Problem 1 instance S into a Problem 2 instance S'.
- Solution map T: transforms solution f' for Problem 2 instance S' into a solution f for Problem 1 instance S.

Reductions

Suppose we have a good algorithm $\mathcal{A}$ for learning binary classifiers.

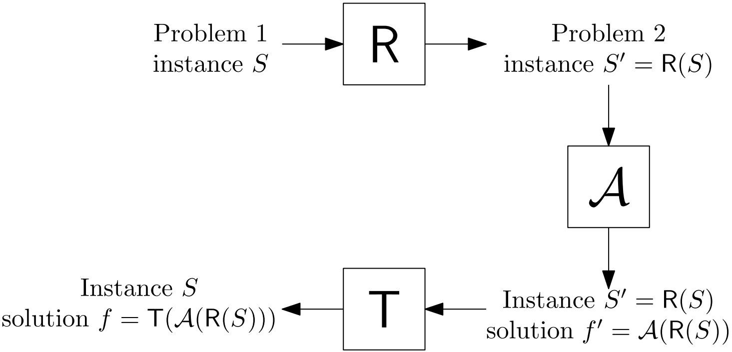
Question: Can we use $\mathcal{A}$ to solve other machine learning problems?

A **reduction** from Problem 1 to Problem 2 is specified by two procedures:

- Instance map R: transforms Problem 1 instance S into a Problem 2 instance S'.
- Solution map T: transforms solution f' for Problem 2 instance S' into a solution f for Problem 1 instance S.

Desired property:
If f' is a good solution for S', then f is a good solution for S.

Reductions



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 function
- $\mathcal{A}$: the latest, greatest learning algorithm for Problem 2

Examples

1. **Problem**: importance-weighted classifiation
   - **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.)

Importance-weighted classification

Problem:

- **Setting**: Random triple $(X, Y, W) \sim P$ for some probability distribution P over $\mathcal{X} \times \mathcal{Y} \times \mathbb{R}_+$.

  $W =$ **importance weight** for labeled example (X, Y).
- **Goal**: Function $f \colon \mathcal{X} \to \mathcal{Y}$ with small **importance-weighted error**:

$$\mathbb{E}\Big[W \cdot \mathbb{1}\{f(X) \neq Y\}\Big].$$

Importance-weighted classification

Problem:

- **Setting**: Random triple $(X, Y, W) \sim P$ for some probability distribution P over $\mathcal{X} \times \mathcal{Y} \times \mathbb{R}_+$.

  W = **importance weight** for labeled example (X, Y).
- **Goal**: Function $f\colon \mathcal{X} \to \mathcal{Y}$ with small **importance-weighted error**:

$$\mathbb{E}\Big[W \cdot \mathbb{1}\{f(X) \neq Y\}\Big].$$

Problem instance:

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

Importance-weighted classification

Problem:

- **Setting**: Random triple $(X, Y, W) \sim P$ for some probability distribution P over $\mathcal{X} \times \mathcal{Y} \times \mathbb{R}_+$.

  W = **importance weight** for labeled example (X, Y).
- **Goal**: Function $f \colon \mathcal{X} \to \mathcal{Y}$ with small **importance-weighted error**:

$$\mathbb{E}\Big[W \cdot \mathbb{1}\{f(X) \neq Y\}\Big].$$

Problem instance:

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

Where it comes up:

- *Class-specific weights*: e.g., $W = 100 \Leftrightarrow Y = 0$ (and $W = 1$ otherwise).
- *Covariate-specific weights*: e.g., $W = 100 \Leftrightarrow X \in \mathcal{X}_0$ (and $W = 1$ o.w.).
- *Boosting*, *domain adaptation*, *causal inference*, ...

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

Importance-weighted classification

Problem:

- **Setting**: Random triple $(X, Y, W) \sim P$ for some probability distribution P over $\mathcal{X} \times \mathcal{Y} \times \mathbb{R}_+$.

  W = **importance weight** for labeled example (X, Y).
- **Goal**: Function $f \colon \mathcal{X} \to \mathcal{Y}$ with small **importance-weighted error**:

$$\mathbb{E}\Big[W \cdot \mathbb{1}\{f(X) \neq Y\}\Big].$$

Problem instance:

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

Where it comes up:

- *Class-specific weights*: e.g., $W = 100 \Leftrightarrow Y = 0$ (and $W = 1$ otherwise).
- *Covariate-specific weights*: e.g., $W = 100 \Leftrightarrow X \in \mathcal{X}_0$ (and $W = 1$ 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

$$\mathbb{E}_{(X,Y,W)\sim P}\Big[W \cdot \mathbb{1}\{f(X) \neq Y\}\Big] \;=\; \mathbb{E}_{(X',Y')\sim P'}\Big[\mathbb{1}\{f(X') \neq Y'\}\Big].$$

THE REJECTION SAMPLING REDUCTION

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

$$\mathbb{E}_{(X,Y,W)\sim P}\Big[W \cdot \mathbb{1}\{f(X) \neq Y\}\Big] \;=\; \mathbb{E}_{(X',Y')\sim P'}\Big[\mathbb{1}\{f(X') \neq Y'\}\Big].$$

Instance mapping procedure

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

1: Initialize $S' = \emptyset$.

2: Let $w_{\max} := \max_{(x,y,w)\in S} w$.

3: **for** each $(x, y, w) \in S$ **do**

4: Toss a coin with heads bias $\frac{w}{w_{\max}}$.

5: If heads, keep example—put (x, y) into S'.

6: If tails, discard example.

7: **end for**

8: **return** Training data S' from $\mathcal{X} \times \mathcal{Y}$.

The rejection sampling reduction

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

$$\mathbb{E}_{(X,Y,W)\sim P}\Big[W \cdot \mathbb{1}\{f(X) \neq Y\}\Big] \;=\; \mathbb{E}_{(X',Y')\sim P'}\Big[\mathbb{1}\{f(X') \neq Y'\}\Big].$$

Instance mapping procedure

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

```
Initialize S' = ∅.
Let w_max := max_{(x,y,w)∈S} w.
for each (x, y, w) ∈ S do
    Toss a coin with heads bias w / w_max.
    If heads, keep example—put (x, y) into S'.
    If tails, discard example.
end for
return Training data S' from X × Y.
```

Solution mapping procedure: identity map

The rejection sampling reduction

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

Define random variable

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

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

The rejection sampling reduction

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

Define random variable

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

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

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

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

Define random variable

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

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

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

Moreover,

$$\mathbb{E}\left[Q \cdot \mathbb{1}\{f(X) \neq Y\}\right]$$

The rejection sampling reduction

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

Define random variable

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

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

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

Moreover,

$$\mathbb{E}\Big[Q \cdot \mathbb{1}\{f(X) \neq Y\}\Big] \quad = \quad \mathbb{E}\Big[\mathbb{E}\Big[Q \cdot \mathbb{1}\{f(X) \neq Y\} \,\Big|\, (X, Y, W)\Big]\Big]$$

The rejection sampling reduction

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

Define random variable

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

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

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

Moreover,

$$\begin{aligned}
\mathbb{E}\Big[Q \cdot \mathbb{1}\{f(X) \neq Y\}\Big] &= \mathbb{E}\Big[\mathbb{E}\Big[Q \cdot \mathbb{1}\{f(X) \neq Y\} \,\Big|\, (X, Y, W)\Big]\Big] \\
&= \mathbb{E}\Big[W \cdot \mathbb{1}\{f(X) \neq Y\}\Big]
\end{aligned}$$

The rejection sampling reduction

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

Define random variable

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

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

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

Moreover,

$$\begin{aligned} \mathbb{E}\Big[Q \cdot \mathbb{1}\{f(X) \neq Y\}\Big] &= \mathbb{E}\Big[\mathbb{E}\Big[Q \cdot \mathbb{1}\{f(X) \neq Y\} \,\Big|\, (X,Y,W)\Big]\Big] \\ &= \mathbb{E}\Big[W \cdot \mathbb{1}\{f(X) \neq Y\}\Big] \end{aligned}$$

Conclusion:
Prediction error w.r.t. P' = 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 $f \colon \mathcal{X} \to \mathcal{Y}$ with small prediction error $\Pr(f(X) \neq Y)$.

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 $f \colon \mathcal{X} \to \mathcal{Y}$ with small prediction error $\Pr(f(X) \neq Y)$.

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.

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 $f \colon \mathcal{X} \to \mathcal{Y}$ with small prediction error $\Pr(f(X) \neq Y)$.

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

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

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

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

One-Against-All reduction

Instance mapping procedure

Input Training data S from $\mathcal{X} \times \{1, 2, \ldots, K\}$.

Initialize empty sets $S'_1, S'_2, \ldots, S'_k$.
for each $(x, y) \in S$ **do**
    for each $i = 1, 2, \ldots, K$ **do**
        Put $(x, \mathbb{1}\{y = i\}) \in \mathcal{X} \times \{0, 1\}$ into S'_i.
    end for
end for
return Training data sets $S'_1, S'_2, \ldots, S'_K$ from $\mathcal{X} \times \{0, 1\}$.

One-Against-All reduction

Instance mapping procedure

Input Training data S from $\mathcal{X} \times \{1, 2, \ldots, K\}$.

```
Initialize empty sets S'_1, S'_2, ..., S'_k.
for each (x, y) ∈ S do
    for each i = 1, 2, ..., K do
        Put (x, 1{y = i}) ∈ X × {0, 1} into S'_i.
    end for
end for
return  Training data sets S'_1, S'_2, ..., S'_K from X × {0, 1}.
```

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

Solution mapping procedure

Input K binary predictors $f'_1, f'_2, \ldots, f'_K \colon \mathcal{X} \to \{0, 1\}$.

return Function $f \colon \mathcal{X} \to \{1, 2, \ldots, K\}$ where

$$f(x) = \underset{i \in \{1,2,\ldots,K\}}{\arg\max}\ f'_i(x) \quad \text{(breaking ties arbitrarily).}$$

Problem with OAA

OAA multi-class predictor:

$$f(x) = \operatorname*{arg\,max}_{i \in \{1,2,\ldots,K\}} f_i'(x).$$

Problem with OAA

OAA multi-class predictor:

$$f(x) = \underset{i \in \{1,2,\ldots,K\}}{\arg\max}\, f_i'(x).$$

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!)

Problem with OAA

OAA multi-class predictor:

$$f(x) = \underset{i \in \{1,2,\ldots,K\}}{\arg\max}\, f_i'(x).$$

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!)

Common solution: use conditional probability estimation

$$f_i'(x) = \text{estimate of } \Pr[Y = i \mid X = x].$$

Error correcting output codes

Main idea: Create m binary classification problems

given $x \in \mathcal{X}$, predict whether or not $y \in L_i$

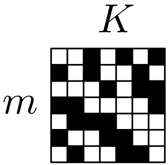
for m pre-specified subsets $L_i \subset \{1, 2, \ldots, K\}$.

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):



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

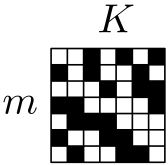
Error correcting output codes

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):



Let i-th column $\boldsymbol{c}_i \in \{0, 1\}^m$ 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 correcting output codes

Instance mapping procedure: similar to OAA.

Error correcting output codes

Instance mapping procedure: similar to OAA.

Solution mapping procedure:

Input m binary predictors $f_1', f_2', \ldots, f_m' \colon \mathcal{X} \to \{0,1\}$.
return Function $f \colon \mathcal{X} \to \{1,2,\ldots,K\}$ where

$$f(x) = \underset{i \in \{1,2,\ldots,K\}}{\arg\min} \left\| \begin{bmatrix} f_1'(x) \\ f_m'(x) \\ \vdots \\ f_m'(x) \end{bmatrix} - \boldsymbol{c}_i \right\|.$$

Error correcting output codes

Instance mapping procedure: similar to OAA.

Solution mapping procedure:

Input m binary predictors $f'_1, f'_2, \ldots, f'_m \colon \mathcal{X} \to \{0,1\}$.
  return Function $f \colon \mathcal{X} \to \{1, 2, \ldots, K\}$ where

$$f(x) = \underset{i \in \{1,2,\ldots,K\}}{\arg\min} \left\| \begin{bmatrix} f'_1(x) \\ f'_m(x) \\ \vdots \\ f'_m(x) \end{bmatrix} - \boldsymbol{c}_i \right\|.$$

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

Error correcting output codes

Instance mapping procedure: similar to OAA.

Solution mapping procedure:

Input m binary predictors $f'_1, f'_2, \dots, f'_m \colon \mathcal{X} \to \{0, 1\}$.
return Function $f \colon \mathcal{X} \to \{1, 2, \dots, K\}$ where

$$f(x) = \operatorname*{arg\,min}_{i \in \{1,2,\dots,K\}} \left\| \begin{bmatrix} f'_1(x) \\ f'_m(x) \\ \vdots \\ f'_m(x) \end{bmatrix} - \boldsymbol{c}_i \right\|.$$

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

Still also useful to use conditional probability estimation

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

("Probabilistic ECOC").

Error correcting output codes

Instance mapping procedure: similar to OAA.

Solution mapping procedure:

Input m binary predictors $f'_1, f'_2, \ldots, f'_m \colon \mathcal{X} \to \{0,1\}$.
return Function $f \colon \mathcal{X} \to \{1, 2, \ldots, K\}$ where

$$f(x) = \underset{i \in \{1,2,\ldots,K\}}{\arg\min} \left\| \begin{bmatrix} f'_1(x) \\ f'_m(x) \\ \vdots \\ f'_m(x) \end{bmatrix} - \boldsymbol{c}_i \right\|.$$

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

Still also useful to use conditional probability estimation

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

("Probabilistic ECOC").

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

Empirical comparison

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

Data set	Number of classes	One-against-all	ECOC
ecoli	8	0.0985	**0.0517**
glass	6	0.3874	**0.3462**
pendigits	10	0.0985	**0.0517**
satimage	6	0.1679	**0.1376**
soybean	19	0.6580	**0.5993**
splice	3	**0.0642**	0.0699
vowel	11	0.6356	**0.5780**
yeast	10	0.4893	**0.4479**

Recap

- **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 classificaiton and prediction error—can be application-/domain-specific.
- **Very useful tools**:
  - Importance-weighted classification (using reductions)
  - Conditional probability estimation (using certain loss functions)