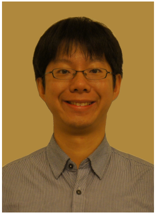
Cost-Sensitive Classification: Algorithms and Advances

Hsuan-Tien Lin
htlin@csie.ntu.edu.tw

Department of Computer Science
& Information Engineering
National Taiwan University

Tutorial for ACML @ Canberra, Australia
November 13, 2013

More about Me



Associate Professor
Dept. CSIE
National Taiwan University

- co-leader of KDDCup world champion teams at NTU: 2010–2013
- research on multi-label classification, ranking, active learning, etc.
- research on cost-sensitive classification: 2007–Present
- Secretary General, Taiwanese Association for Artificial Intelligence
- instructor of Mandarin-teaching MOOC of Machine Learning on NTU-Coursera: 2013.11–

  `https://www.coursera.org/course/ntumlone`

Outline

Is This Your Fingerprint?

?

you

intruder

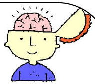
- a **binary classification** problem
  —grouping “fingerprint pictures” into **two** different “categories”

C’mon, we know about
binary classification all too well! :-)

Supervised Machine Learning

parent

(picture, category) pairs

kid's brain  → good decision function

possibilities

truth $f(\mathbf{x})$ + noise $e(\mathbf{x})$

examples (picture $\mathbf{x}_n$, category y_n)

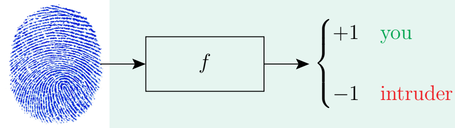
learning algorithm  → good decision function $g(\mathbf{x}) \approx f(\mathbf{x})$

learning model $\{h(\mathbf{x})\}$

how to evaluate whether $g(\mathbf{x}) \approx f(\mathbf{x})$?

Performance Evaluation

Fingerprint Verification



example/figure borrowed from Amazon ML best-seller textbook 

"Learning from Data" 

(Abu-Mostafa, Magdon-Ismail, , 2013)

two types of error: false accept and false reject

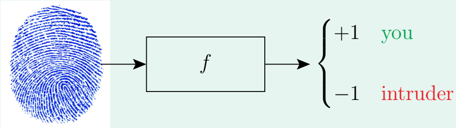
		g: +1	g: -1
f	+1	no error	false reject
	-1	false accept	no error

		g: +1	g: -1
f	+1	0	1
	-1	1	0

simplest choice:
penalizes both types **equally** and calculate **average** penalties

Fingerprint Verification for Supermarket

Fingerprint Verification



two types of error: false accept and false reject

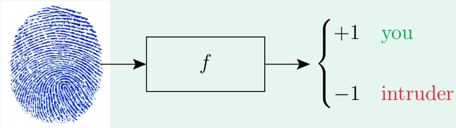
		g: +1	g: -1
f	+1	no error	false reject
	-1	false accept	no error

		g: +1	g: -1
f	+1	0	10
	-1	1	0

- supermarket: fingerprint for discount
- false reject: **very unhappy customer, lose future business**
- false accept: give a minor discount, intruder left fingerprint **:-)**

Fingerprint Verification for CIA

Fingerprint Verification



two types of error: false accept and false reject

		g: +1	g: -1
f	+1	no error	false reject
f	-1	false accept	no error

		g: +1	g: -1
f	+1	0	1
f	-1	1000	0

- CIA: fingerprint for entrance
- false accept: **very serious consequences!**
- false reject: unhappy employee, but so what? **:-)**

Regular Binary Classification

penalizes both types equally

		$h(\mathbf{x})$ +1	$h(\mathbf{x})$ -1
y	+1	0	1
	-1	1	0

in-sample error for any hypothesis h

$$E_{\text{in}}(h) = \frac{1}{N} \left[\!\left[\underbrace{y_n}_{f(\mathbf{x}_n)+\text{noise}} \neq h(\mathbf{x}_n) \right]\!\right]$$

out-of-sample error for any hypothesis h

$$E_{\text{out}}(h) = \underset{(\mathbf{x},y)}{\mathcal{E}} \left[\!\left[\underbrace{y}_{f(\mathbf{x})+\text{noise}} \neq h(\mathbf{x}) \right]\!\right]$$

regular binary classification:
well-studied in machine learning
—**ya, we know! :-)**

Class-Weighted Cost-Sensitive Binary Classification

Supermarket Cost (Error, Loss, ...) Matrix

		$h(\mathbf{x})$ +1	$h(\mathbf{x})$ -1
y	+1	0	10
	-1	1	0

in-sample

$$E_{\text{in}}(h) = \frac{1}{N}\sum_{n=1}^{N} \begin{Bmatrix} 10 & \text{if } y_n = +1 \\ 1 & \text{if } y_n = -1 \end{Bmatrix} \cdot [\![y_n \neq h(\mathbf{x}_n)]\!]$$

out-of-sample

$$E_{\text{out}}(h) = \mathop{\mathcal{E}}_{(\mathbf{x},y)} \begin{Bmatrix} 10 & \text{if } y = +1 \\ 1 & \text{if } y = -1 \end{Bmatrix} \cdot [\![y \neq h(\mathbf{x})]\!]$$

class-weighted cost-sensitive binary classification: different 'weight' for different y

Setup: Class-Weighted Cost-Sensitive Binary Classification

Given

N examples, each

(input $\mathbf{x}_n$, label y_n) $\in \mathcal{X} \times \{-1, +1\}$

and weights w_+, w_-

		$h(\mathbf{x})$ +1	$h(\mathbf{x})$ -1
y	+1	0	w_+
y	-1	w_-	0

representing the two entries of the cost matrix

Goal

a classifier $g(\mathbf{x})$ that

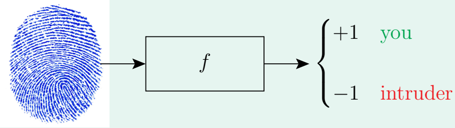
pays a small cost $w_y \, [\![y \neq g(\mathbf{x})]\!]$

on future **unseen** example $(\mathbf{x}, y)$, i.e., achieves low $E_{\text{out}}(g)$

regular classification: $w_+ = w_-$ (= 1)

Supermarket Revisited

Fingerprint Verification



two types of error: false accept and false reject

		g: +1	g: -1
f	+1	no error	false reject
	-1	false accept	no error

		g: +1	g: -1
f	big customer	0	100
	usual customer	0	10
	intruder	1	0

- supermarket: fingerprint for discount
- big customer: really don't want to lose her/his business
- usual customer: don't want to lose business, but not so serious

Example-Weighted Cost-Sensitive Binary Classification

Supermarket Cost Vectors (Rows)

		$h(\mathbf{x})$ +1	$h(\mathbf{x})$ -1
y	big	0	100
	usual	0	10
	intruder	1	0

in-sample

$$E_{\text{in}}(h) = \frac{1}{N}\sum_{n=1}^{N} \underbrace{w_n}_{\text{importance}} \cdot [\![y_n \neq h(\mathbf{x}_n)]\!]$$

out-of-sample

$$E_{\text{out}}(h) = \underset{(\mathbf{x},y,w)}{\mathcal{E}} w \cdot [\![y \neq h(\mathbf{x})]\!]$$

example-weighted cost-sensitive binary classification:
different w for different $(\mathbf{x}, y)$
—seen this in AdaBoost? :-)

Setup: Example-Weighted Cost-Sensitive Binary Classification

Given

N examples, each (input $\mathbf{x}_n$, label y_n) $\in \mathcal{X} \times \{-1, +1\}$

and weight $w_n \in \mathbb{R}^+$

Goal

a classifier $g(\mathbf{x})$ that

pays a small cost $w \, [\![y \neq g(\mathbf{x})]\!]$

on future **unseen** example $(\mathbf{x}, y, w)$, i.e., achieves low $E_{\text{out}}(g)$

regular $\subset$ class-weighted $\subset$ example-weighted

Outline

Key Idea: Conditional Probability Estimator

Goal (Class-Weighted Setup)

a classifier $g(\mathbf{x})$ that pays a small cost $w_y \llbracket y \neq g(\mathbf{x}) \rrbracket$ on future **unseen** example $(\mathbf{x}, y)$

- expected error for predicting +1 on $\mathbf{x}$: $w_- P(-1|\mathbf{x})$
- expected error for predicting -1 on $\mathbf{x}$: $w_+ P(+1|\mathbf{x})$

if $P(y|\mathbf{x})$ known

Bayes optimal $g^*(\mathbf{x}) =$

$$\text{sign}\Big(w_+ P(+1|\mathbf{x}) - w_- P(-1|\mathbf{x})\Big)$$

if $p(\mathbf{x}) \approx P(+1|\mathbf{x})$ well

approximately good $g_p(\mathbf{x}) =$

$$\text{sign}\Big(w_+ p(\mathbf{x}) - w_-(1 - p(\mathbf{x}))\Big)$$

how to get conditional probability estimator p?
logistic regression, Naïve Bayes, . . .

Approximate Bayes-Optimal Decision

if $p(\mathbf{x}) \approx P(+1|\mathbf{x})$ well

$$\text{approximately good } g_p(\mathbf{x}) = \text{sign}\Big(w_+ p(\mathbf{x}) - w_-(1 - p(\mathbf{x}))\Big)$$

that is (Elkan, 2001),

$$g_p(\mathbf{x}) = +1 \text{ iff } \quad w_+ p(\mathbf{x}) - w_-(1 - p(\mathbf{x})) > 0$$

$$\text{iff} \quad p(\mathbf{x}) > \frac{w_-}{w_+ + w_-}$$

: $\frac{1}{11}$ for supermarket; $\frac{100}{101}$ for CIA

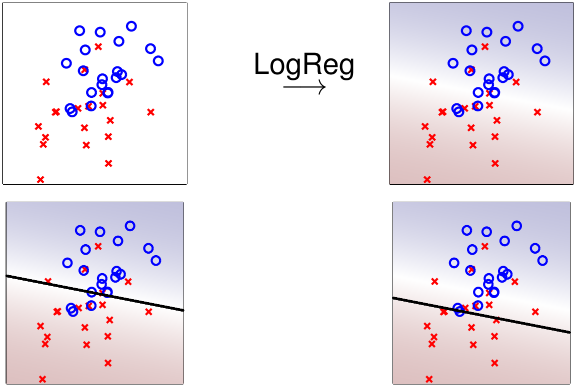
Approximate Bayes-Optimal Decision (ABOD) Approach

1. use your favorite algorithm on $\{(\mathbf{x}_n, y_n)\}$ to get $p(\mathbf{x}) \approx P(+1|\mathbf{x})$
2. for each new input $\mathbf{x}$, predict its class using $g_p(\mathbf{x}) = \text{sign}\left(p(\mathbf{x}) - \frac{w_-}{w_+ + w_-}\right)$

'simplest' approach:
probability estimate + threshold changing

ABOD on Artificial Data

1. use your favorite algorithm on $\{(\mathbf{x}_n, y_n)\}$ to get $p(\mathbf{x}) \approx P(+1|\mathbf{x})$
2. for each new input $\mathbf{x}$, predict its class using
   $g_p(\mathbf{x}) = \text{sign}\left(p(\mathbf{x}) - \frac{w_-}{w_+ + w_-}\right)$



regular — supermarket

		g	
		+1	-1
y	+1	0	10
	-1	1	0

Pros and Cons of ABOD

Pros

- optimal: if good probability estimate: $p(\mathbf{x})$ really close to $P(+1|\mathbf{x})$
- simple: training (probability estimate) unchanged, and prediction (threshold) changed only a little

Cons

- 'difficult': good probability estimate often more difficult than good binary classification
- 'restricted': only applicable to class-weighted setup —need 'full picture' of cost matrix

approach for the example-weighted setup?

Outline

Key Idea: Example Weight = Copying

Goal

a classifier $g(\mathbf{x})$ that

pays a small cost $w \llbracket y \neq g(\mathbf{x}) \rrbracket$

on future **unseen** example $(\mathbf{x}, y, w)$

on one $(\mathbf{x}, y)$

wrong prediction charged by w

on w copies of $(\mathbf{x}, y)$

wrong prediction charged by 1
—regular classification

how to copy? **over-sampling**

Example-Weighted Classification by Over-Sampling

copy each $(\mathbf{x}_n, y_n)$ for w_n times

original problem

evaluate with

		$h(\mathbf{x})$ +1	$h(\mathbf{x})$ -1
y	big	0	100
	usual	0	10
	intruder	1	0

$(\mathbf{x}_1, -1, 1)$
$(\mathbf{x}_2, +1, 10)$
$(\mathbf{x}_3, +1, 100)$
$(\mathbf{x}_4, +1, 10)$
$(\mathbf{x}_5, -1, 1)$

equivalent problem

evaluate with

		$h(\mathbf{x})$ +1	$h(\mathbf{x})$ -1
y	big	0	1
	usual	0	1
	intruder	1	0

$(\mathbf{x}_1, -1)$
$(\mathbf{x}_2, +1), \ldots, (\mathbf{x}_2, +1)$
$(\mathbf{x}_3, +1), \ldots, (\mathbf{x}_3, +1), \ldots, (\mathbf{x}_3, +1)$
$(\mathbf{x}_4, +1), \ldots, (\mathbf{x}_4, +1)$
$(\mathbf{x}_5, -1)$

how to learn a good g for RHS?
SVM, NNet, ...

Cost-Proportionate Example Weighting

Cost-Proportionate Example Weighting (CPEW) Approach

1. effectively transform $\{(\mathbf{x}_n, y_n, w_n)\}$ to $\{(\mathbf{x}_m, y_m)\}$ such that the 'copies' of $(\mathbf{x}_n, y_n)$ in $\{(\mathbf{x}_m, y_m)\}$ is proportional to w_n
   - over/under-sampling with normalized w_n (Elkan, 2001)
   - under-sampling by rejection (Zadrozny, 2003)
   - modify existing algorithms equivalently (Zadrozny, 2003)
2. use your favorite algorithm on $\{(\mathbf{x}_m, y_m)\}$ to get binary classifier $g(\mathbf{x})$
3. for each new input $\mathbf{x}$, predict its class using $g(\mathbf{x})$

simple and general:
very popular for cost-sensitive binary classification

CPEW by Modification

1. effectively transform $\{(\mathbf{x}_n, y_n, w_n)\}$ to $\{(\mathbf{x}_m, y_m)\}$ such that the 'copies' of $(\mathbf{x}_n, y_n)$ in $\{(\mathbf{x}_m, y_m)\}$ is proportional to w_n
   - modify existing algorithms equivalently (Zadrozny, 2003)
2. use your favorite algorithm on $\{(\mathbf{x}_m, y_m)\}$ to get binary classifier $g(\mathbf{x})$
3. for each new input $\mathbf{x}$, predict its class using $g(\mathbf{x})$

Regular Linear SVM

$$\min_{\mathbf{w},b} \quad \frac{1}{2}\langle \mathbf{w}, \mathbf{w}\rangle + \sum_{n=1}^{N} C\xi_n$$

$$\xi_n = \max\left(1 - y_n(\langle \mathbf{w}, \mathbf{x}_n\rangle + b), 0\right)$$

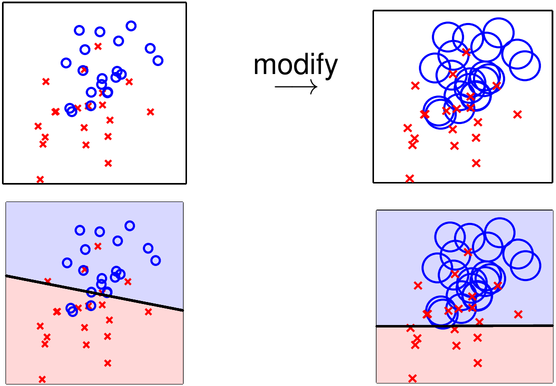
Modified Linear SVM

$$\min_{\mathbf{w},b} \quad \frac{1}{2}\langle \mathbf{w}, \mathbf{w}\rangle + \sum_{n=1}^{N} C \cdot w_n \cdot \xi_n$$

$$\xi_n = \max\left(1 - y_n(\langle \mathbf{w}, \mathbf{x}_n\rangle + b), 0\right)$$

CPEW by Modification on Artificial Data

1. effectively transform $\{(\mathbf{x}_n, y_n, w_n)\}$ to $\{(\mathbf{x}_m, y_m)\}$ by modifying existing algorithms equivalently (Zadrozny, 2003)
2. use your favorite algorithm on $\{(\mathbf{x}_m, y_m)\}$ to get $g(\mathbf{x})$
3. for each new input $\mathbf{x}$, predict its class using $g(\mathbf{x})$



regular | supermarket

		g: +1	g: -1
y	+1	0	10
	-1	1	0

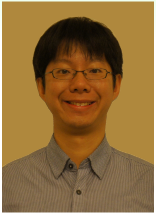
CPEW by Rejection Sampling

COSTING Algorithm (Zadrozny, 2003)

1. effectively transform $\{(\mathbf{x}_n, y_n, w_n)\}$ to $\{(\mathbf{x}_m, y_m)\}$ such that the 'copies' of $(\mathbf{x}_n, y_n)$ in $\{(\mathbf{x}_m, y_m)\}$ is proportional to w_n
   - under-sampling by rejection (Zadrozny, 2003)
2. use your favorite algorithm on $\{(\mathbf{x}_m, y_m)\}$ to get binary classifier $g(\mathbf{x})$
3. repeat (1) and (2) to get multiple g and aggregate them
4. for each new input $\mathbf{x}$, predict its class using aggregated $g(\mathbf{x})$

commonly used when your favorite algorithm is a black box rather than a white box

Biased Personal Favorites



- CPEW by Modification if possible
- COSTING: fast training and stable performance
- ABOD if in the mood for Bayesian :-)

Outline

Which Digit Did You Write?

?

one (1) two (2) three (3) four (4)

- a **multiclass classification** problem
  —grouping “pictures” into different “categories”

C’mon, we know about
multiclass classification all too well! :-)

Performance Evaluation ($g(\mathbf{x}) \approx f(\mathbf{x})$?)

?

- ZIP code recognition:
  1: wrong; 2: right; 3: wrong; 4: wrong
- check value recognition:
  1: one-dollar mistake; 2: no mistake;
  3: one-dollar mistake; 4: **two**-dollar mistake
- evaluation by formation similarity:
  1: not very similar; 2: very similar;
  3: somewhat similar; 4: a **silly** prediction

different applications:
evaluate mis-predictions differently

ZIP Code Recognition

?

1: wrong; 2: right; 3: wrong; 4: wrong

- **regular** multiclass classification: only right or wrong
- wrong cost: 1; right cost: 0
- prediction error of h on some $(\mathbf{x}, y)$:

$$\text{classification cost} = [\![y \neq h(\mathbf{x})]\!]$$

—as discussed in regular binary classification

regular multiclass classification:
well-studied, many good algorithms

Check Value Recognition

?

1: one-dollar mistake; 2: no mistake;
3: one-dollar mistake; 4: **two**-dollar mistake

- **cost-sensitive** multiclass classification: different costs for different mis-predictions
- e.g. prediction error of h on some $(\mathbf{x}, y)$:

$$\text{absolute cost} = |y - h(\mathbf{x})|$$

next: cost-sensitive **multiclass** classification

What is the Status of the Patient?

?

H1N1-infected

cold-infected

healthy

- another **classification** problem
  —grouping “patients” into different “status”

are all mis-prediction costs equal?

Patient Status Prediction

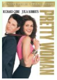
error measure = society cost

actual \ predicted	H7N9	cold	healthy
H7N9	0	1000	**100000**
cold	100	0	3000
healthy	100	30	0

- H7N9 mis-predicted as healthy: **very high cost**
- cold mis-predicted as healthy: high cost
- cold correctly predicted as cold: no cost

human doctors consider costs of decision;
can computer-aided diagnosis do the same?

What is the Type of the Movie?

			
?	romance	fiction	terror

customer 1 who hates romance but likes terror

error measure = non-satisfaction

actual \ predicted	romance	fiction	terror
romance	0	5	100

customer 2 who likes terror and romance

actual \ predicted	romance	fiction	terror
romance	0	5	3

different customers:

evaluate mis-predictions differently

Cost-Sensitive Multiclass Classification Tasks

movie classification with non-satisfaction

actual \ predicted	romance	fiction	terror
customer 1, romance	0	5	100
customer 2, romance	0	5	3

patient diagnosis with society cost

actual \ predicted	H7N9	cold	healthy
H7N9	0	1000	**100000**
cold	100	0	3000
healthy	100	30	0

check digit recognition with absolute cost

$$\mathcal{C}(y, h(\mathbf{x})) = |y - h(\mathbf{x})|$$

Cost Vector

cost vector $\mathbf{c}$: a row of cost components

- customer 1 on a romance movie: $\mathbf{c} = (0, 5, 100)$
- an H7N9 patient: $\mathbf{c} = (0, 1000, 100000)$
- absolute cost for digit 2: $\mathbf{c} = (1, 0, 1, 2)$
- "regular" classification cost for label 2: $\mathbf{c}_c^{(2)} = (1, 0, 1, 1)$

regular classification:
special case of cost-sensitive classification

Setup: Matrix-Based Cost-Sensitive Binary Classification

Given

N examples, each (input $\mathbf{x}_n$, label y_n) $\in \mathcal{X} \times \{1, 2, \ldots, K\}$

and cost matrix $\mathcal{C} \in \mathbb{R}^{K \times K}$

—will assume $\mathcal{C}(y, y) = 0 = \min_{1 \le k \le K} \mathcal{C}(y, k)$

Goal

a classifier $g(\mathbf{x})$ that

pays a small cost $\mathcal{C}(y, g(\mathbf{x}))$

on future **unseen** example $(\mathbf{x}, y)$

extension of 'class-weighted' cost-sensitive binary classification

Setup: Vector-Based Cost-Sensitive Binary Classification

Given

N examples, each (input $\mathbf{x}_n$, label y_n) $\in \mathcal{X} \times \{1, 2, \ldots, K\}$

and cost vector $\mathbf{c}_n \in \mathbb{R}^K$

—will assume $\mathbf{c}_n[y_n] = 0 = \min_{1 \le k \le K} \mathbf{c}_n[k]$

Goal

a classifier $g(\mathbf{x})$ that pays a small cost $\mathbf{c}[g(\mathbf{x})]$ on future **unseen** example $(\mathbf{x}, y, \mathbf{c})$

- will assume $\mathbf{c}[y] = 0 = c_{\min} = \min_{1 \le k \le K} \mathbf{c}[k]$
- note: y not really needed in evaluation

extension of 'example-weighted'
cost-sensitive binary classification

Which Age-Group?

?

infant (1)

child (2)

teen (3)

adult (4)

- small mistake—classify a child as a teen;
  big mistake—classify an infant as an adult
- cost matrix $\mathcal{C}(y, g(x))$ for embedding 'order': $\mathcal{C} = \begin{pmatrix} 0 & 1 & 4 & 5 \\ 1 & 0 & 1 & 3 \\ 3 & 1 & 0 & 2 \\ 5 & 4 & 1 & 0 \end{pmatrix}$

cost-sensitive classification can help solve many other problems, such as **ordinal ranking**

Outline

Key Idea: Conditional Probability Estimator

Goal (Matrix Setup)

a classifier $g(\mathbf{x})$ that pays a small cost $\mathcal{C}(y, g(\mathbf{x}))$ on future **unseen** example $(\mathbf{x}, y)$

if $P(y|\mathbf{x})$ known

Bayes optimal $g^*(\mathbf{x}) =$

$$\underset{1 \le k \le K}{\operatorname{argmin}} \sum_{y=1}^{K} P(y|\mathbf{x}) \mathcal{C}(y, k)$$

if $p(y, \mathbf{x}) \approx P(y|\mathbf{x})$ well

approximately good $g_p(\mathbf{x}) =$

$$\underset{1 \le k \le K}{\operatorname{argmin}} \sum_{y=1}^{K} p(y, \mathbf{x}) \mathcal{C}(y, k)$$

how to get conditional probability estimator p?
logistic regression, Naïve Bayes, . . .

Approximate Bayes-Optimal Decision

if $p(y, \mathbf{x}) \approx P(+1|\mathbf{x})$ well

(Domingos, 1999)

approximately good $g_p(\mathbf{x}) = \text{argmin}_{k \in \{1,2,\ldots,K\}} \sum_{y=1}^{K} p(y, \mathbf{x}) \mathcal{C}(y, k)$

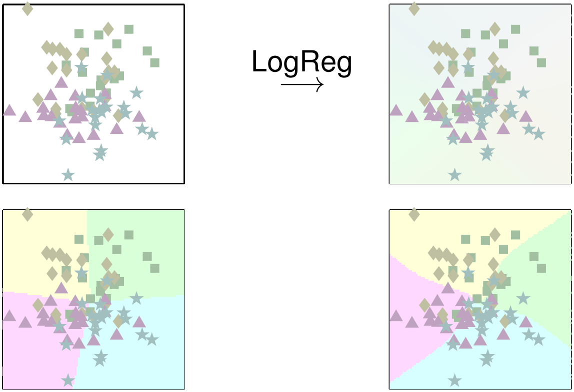
Approximate Bayes-Optimal Decision (ABOD) Approach

1. use your favorite algorithm on $\{(\mathbf{x}_n, y_n)\}$ to get $p(y, \mathbf{x}) \approx P(y|\mathbf{x})$
2. for each new input $\mathbf{x}$, predict its class using $g_p(\mathbf{x})$ above

a simple extension from binary classification:
probability estimate + Bayes-optimal decision

ABOD on Artificial Data

1. use your favorite algorithm on $\{(\mathbf{x}_n, y_n)\}$ to get $p(y, \mathbf{x}) \approx P(y|\mathbf{x})$
2. for each new input $\mathbf{x}$, predict its class using $g_p(\mathbf{x})$



regular　　　rotate

		g = 1	2	3	4
y	1	0	1	2	4
	2	4	0	1	2
	3	2	4	0	1
	4	1	2	4	0

Pros and Cons of ABOD

Pros

- optimal: if good probability estimate: $p(y, \mathbf{x})$ really close to $P(y|\mathbf{x})$
- simple: with training (probability estimate) unchanged, and prediction (threshold) changed only a little

Cons

- 'difficult': good probability estimate often more difficult than good multiclass classification
- 'restricted': only applicable to class-weighted setup —need 'full picture' of cost matrix
- 'slow prediction': need sophisticated calculation at prediction stage

can we use any multiclass classification algorithm for ABOD?

MetaCost Approach

Approximate Bayes-Optimal Decision (ABOD) Approach

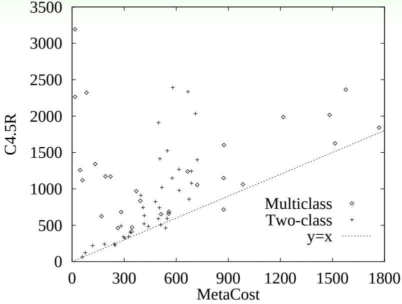
1. use your favorite algorithm on $\{(\mathbf{x}_n, y_n)\}$ to get $p(y, \mathbf{x}) \approx P(y|\mathbf{x})$
2. for each new input $\mathbf{x}$, predict its class using $g_p(\mathbf{x})$

MetaCost Approach (Domingos, 1999)

1. use your favorite multiclass classification algorithm on bootstrapped $\{(\mathbf{x}_n, y_n)\}$ and aggregate the classifiers to get $p(y, \mathbf{x}) \approx P(y|\mathbf{x})$
2. for each given input $\mathbf{x}_n$, relabel it to y'_n using $g_p(\mathbf{x})$
3. run your favorite multiclass classification algorithm on relabeled $\{(\mathbf{x}_n, y'_n)\}$ to get final classifier g
4. for each new input $\mathbf{x}$, predict its class using $g(\mathbf{x})$

pros: any multiclass classification algorithm can be used

MetaCost on Semi-Real Data



(Domingos, 1999)

- some "random" cost with UCI data
- MetaCost+C4.5: cost-sensitive
- C4.5: regular

not surprisingly,

considering the cost properly does help

Outline

Recall: Example-Weighting Useful for Binary

can example weighting be used for multiclass?

Yes! an elegant solution if using cost matrix with special properties (Zhou, 2010)

$$\frac{\mathcal{C}(i,j)}{\mathcal{C}(j,i)} = \frac{w_i}{w_j}$$

what if using cost vectors without special properties?

Key Idea: Cost Transformation

$$\underbrace{\begin{pmatrix} 0 & 1000 \end{pmatrix}}_{\mathbf{c}} = \underbrace{\begin{pmatrix} 1000 & 0 \end{pmatrix}}_{\#\text{ of copies}} \cdot \underbrace{\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}}_{\text{classification costs}}$$

$$\underbrace{\begin{pmatrix} 3 & 2 & 3 & 4 \end{pmatrix}}_{\text{cost } \mathbf{c}} = \underbrace{\begin{pmatrix} 1 & 2 & 1 & 0 \end{pmatrix}}_{\text{mixture weights } q_\ell} \cdot \underbrace{\begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}}_{\text{classification costs}}$$

- **split** the cost-sensitive example: $(\mathbf{x}, 2)$ with $\mathbf{c} = (3, 2, 3, 4)$ equivalent to a weighted mixture $\{(\mathbf{x}, 1, 1), (\mathbf{x}, 2, 2), (\mathbf{x}, 3, 1)\}$

cost equivalence: $\mathbf{c}[h(\mathbf{x})] = \sum_{\ell=1}^{K} q_\ell \left[\!\left[\ell \neq h(\mathbf{x}) \right]\!\right]$ for any h

Meaning of Cost Equivalence

$$\mathbf{c}[h(\mathbf{x})] = \sum_{\ell=1}^{K} q_\ell \, [\![\ell \neq h(\mathbf{x})]\!]$$

on one $(\mathbf{x}, y, \mathbf{c})$

wrong prediction charged by $\mathbf{c}[h(\mathbf{x})]$

on all $(\mathbf{x}, \ell, q_\ell)$

wrong prediction charged by total weighted classification error —weighted classification

weighted classification $\Longrightarrow$ regular classification?
same as binary (with CPEW) when $q_\ell \geq 0$

	$\min_g$ expected LHS	(original cost-sensitive problem)
$=$	$\min_g$ expected RHS	(a regular problem when $q_\ell \geq 0$)

Cost Transformation Methodology: Preliminary

1. split each training example $(\mathbf{x}_n, y_n, \mathbf{c}_n)$ to a weighted mixture $\{(\mathbf{x}_n, \ell, q_{n,\ell})\}_{\ell=1}^{K}$
2. apply regular/weighted classification algorithm on the weighted mixtures $\bigcup_{n=1}^{N} \{(\mathbf{x}_n, \ell, q_{n,\ell})\}_{\ell=1}^{K}$

- by $\mathbf{c}[g(\mathbf{x})] = \sum_{\ell=1}^{K} q_\ell \, [\![\ell \neq g(\mathbf{x})]\!]$ (cost equivalence),
  good g for new regular classification problem
  $=$ good g for original cost-sensitive classification problem
- regular classification: needs $q_{n,\ell} \geq 0$

but what if $q_{n,\ell}$ negative?

Similar Cost Vectors

$$\underbrace{\begin{pmatrix} 1 & 0 & 1 & 2 \\ 3 & 2 & 3 & 4 \end{pmatrix}}_{\text{costs}} = \underbrace{\begin{pmatrix} 1/3 & 4/3 & 1/3 & -2/3 \\ 1 & 2 & 1 & 0 \end{pmatrix}}_{\text{mixture weights } q_\ell} \cdot \underbrace{\begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}}_{\text{classification costs}}$$

- negative q_ℓ: cannot split
- but $\hat{\mathbf{c}} = (1, 0, 1, 2)$ is **similar** to $\mathbf{c} = (3, 2, 3, 4)$: for any classifier g,

$$\hat{\mathbf{c}}[g(\mathbf{x})] + \text{constant} = \mathbf{c}[g(\mathbf{x})] = \sum_{\ell=1}^{K} q_\ell \, [\![\ell \neq g(\mathbf{x})]\!]$$

- constant can be dropped during minimization

$$\begin{array}{rll} & \min_g \text{ expected } \hat{\mathbf{c}}[g(\mathbf{x})] & \text{(original cost-sensitive problem)} \\ = & \min_g \text{ expected LHS} & \text{(a regular problem when } q_\ell \geq 0\text{)} \end{array}$$

Cost Transformation Methodology: Revised

1. shift each training cost $\hat{\mathbf{c}}_n$ to a similar and "splittable" $\mathbf{c}_n$
2. split $(\mathbf{x}_n, y_n, \mathbf{c}_n)$ to a weighted mixture $\{(\mathbf{x}_n, \ell, q_{n,\ell})\}_{\ell=1}^{K}$
3. apply regular classification algorithm on the weighted mixtures $\bigcup_{n=1}^{N} \{(\mathbf{x}_n, \ell, q_{n,\ell})\}_{\ell=1}^{K}$

- **splittable**: $q_{n,\ell} \geq 0$
- by cost equivalence after shifting:

  good g for new regular classification problem

  $=$ good g for original cost-sensitive classification problem

but infinitely many similar and splittable $\mathbf{c}_n$!

Uncertainty in Mixture

- a single example $\{(x, 2)\}$
  —**certain** that the desired label is 2
- a mixture $\{(x, 1, 1), (x, 2, 2), (x, 3, 1)\}$ sharing the same x
  —**uncertainty** in the desired label (25%: 1, 50%: 2, 25%: 3)
- over-shifting adds unnecessary mixture uncertainty:

$$\underbrace{\begin{pmatrix} 3 & 2 & 3 & 4 \\ 33 & 32 & 33 & 34 \end{pmatrix}}_{\text{costs}} = \underbrace{\begin{pmatrix} 1 & 2 & 1 & 0 \\ 11 & 12 & 11 & 10 \end{pmatrix}}_{\text{mixture weights}} \cdot \underbrace{\begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}}_{\text{classification costs}}$$

should choose a similar and splittable **c**
with **minimum mixture uncertainty**

Cost Transformation Methodology: Final

Cost Transformation Methodology (Lin, 2010)

1. shift each training cost $\hat{\mathbf{c}}_n$ to a similar and splittable $\mathbf{c}_n$ with minimum "mixture uncertainty"
2. split $(\mathbf{x}_n, y_n, \mathbf{c}_n)$ to a weighted mixture $\{(\mathbf{x}_n, \ell, q_{n,\ell})\}_{\ell=1}^{K}$
3. apply regular classification algorithm on the weighted mixtures

$$\bigcup_{n=1}^{N} \{(\mathbf{x}_n, \ell, q_{n,\ell})\}_{\ell=1}^{K}$$

- mixture uncertainty: entropy of normalized $(q_1, q_2, \cdots, q_K)$
- a simple and unique optimal shifting exists for every $\hat{\mathbf{c}}$

good g for new regular classification problem
$=$ good g for original cost-sensitive classification problem

Data Space Expansion Approach

Data Space Expansion (DSE) Approach (Abe, 2004)

1. for each $(\mathbf{x}_n, y_n, \mathbf{c}_n)$ and ℓ, let $q_{n,\ell} = \max_{1 \le k \le K} \mathbf{c}_n[k] - \mathbf{c}_n[\ell]$
2. apply your favorite multiclass classification algorithm on the weighted mixtures $\bigcup_{n=1}^{N} \{(\mathbf{x}_n, \ell, q_{n,\ell})\}_{\ell=1}^{K}$ to get $g(\mathbf{x})$
3. for each new input $\mathbf{x}$, predict its class using $g(\mathbf{x})$

- detailed explanation provided by the cost transformation methodology discussed above (Lin, 2010)
- extension of Cost-Proportionate Example Weighting, but now with relabeling!

pros: any multiclass classification algorithm can be used

DSE versus MetaCost on Semi-Real Data

(Abe, 2004)

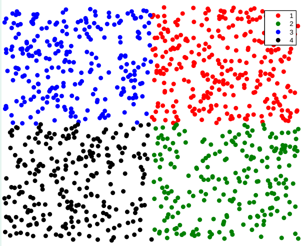
	MetaCost	DSE
annealing	206.8	127.1
solar	5317	110.9
kdd99	49.39	46.68
letter	129.6	114.0
splice	49.95	135.5
satellite	104.4	116.8

- some "random" cost with UCI data
- C4.5 with COSTING for weighted classification

DSE comparable to MetaCost

Cons of DSE: Unavoidable (Minimum) Uncertainty

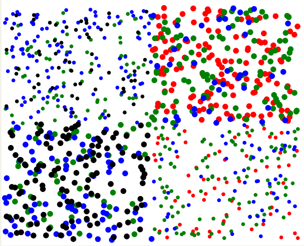
Original Cost-Sensitive Classification Problem



individual examples with certainty

+ absolute cost =

New Regular Classification Problem



mixtures with unavoidable uncertainty

- cost embedded as weight + label
- new problem usually **harder** than original one

need robust multiclass classification algorithm to deal with uncertainty

Outline

Key Idea: Design Robust Multiclass Algorithm

One-Versus-One: A Popular Classification Meta-Method

1. for a pair (i, j), take all examples $(\mathbf{x}_n, y_n)$ that $y_n = i$ or j **(original one-versus-one)**
2. for a pair (i, j), from each weighted mixture $\{(\mathbf{x}_n, \ell, q_{n,\ell})\}$ with $q_{n,i} > q_{n,j}$, take (x_n, i) with weight $q_{n,i} - q_{n,j}$; vice versa **(robust one-versus-one)**
3. train a binary classifier $\hat{g}^{(i,j)}$ using those examples
4. repeat the previous two steps for all different (i, j)
5. predict using the votes from $\hat{g}^{(i,j)}$

- un-shifting inside the meta-method to remove uncertainty
- robust step makes it suitable for cost transformation methodology

cost-sensitive one-versus-one:
cost transformation + robust one-versus-one

Cost-Sensitive One-Versus-One (CSOVO)

Cost-Sensitive One-Versus-One (Lin, 2010)

1. for a pair (i, j), transform all examples $(\mathbf{x}_n, y_n)$ to
$$\left(x_n, \operatorname*{argmin}_{k \in \{i,j\}} \mathbf{c}_n[k]\right) \text{ with weight } |\mathbf{c}_n[i] - \mathbf{c}_n[j]|$$
2. train a binary classifier $\hat{g}^{(i,j)}$ using those examples
3. repeat the previous two steps for all different (i, j)
4. predict using the votes from $\hat{g}^{(i,j)}$

- comes with **good theoretical guarantee**:

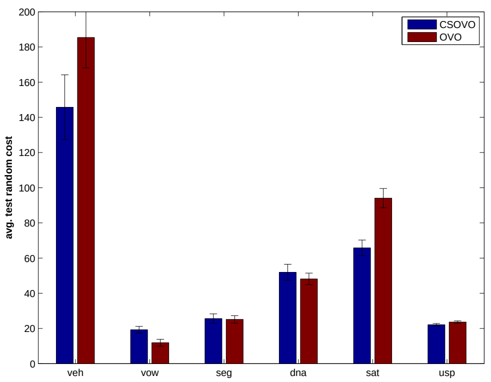
$$\text{test cost of final classifier} \leq 2 \sum_{i<j} \text{test cost of } \hat{g}^{(i,j)}$$

- **simple**, **efficient**, and takes original OVO as **special case**

physical meaning:

each $\hat{g}^{(i,j)}$ answers yes/no question "prefer i or j?"

CSOVO on Semi-Real Data



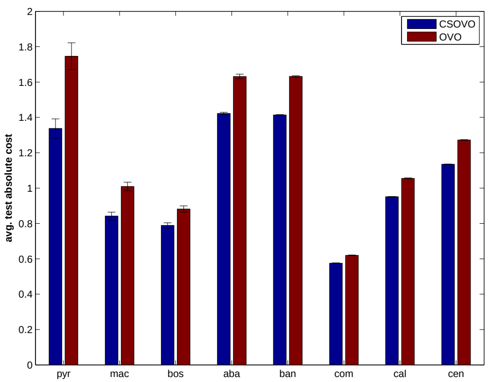
(Lin, 2010)

- some "random" cost with UCI data
- CSOVO-SVM: cost-sensitive
- OVO-SVM: regular

not surprisingly again,

considering the cost properly does help

CSOVO for Ordinal Ranking



(Lin, 2010)

- absolute cost with benchmark ordinal ranking data
- CSOVO-SVM: cost-sensitive
- OVO-SVM: regular

CSOVO significantly better for ordinal ranking

Other Approaches via Weighted Binary Classification

Filter Tree (FT): $K-1$ binary classifiers (Beygelzimer, 2007)

Is the lowest cost within labels $\{1, 4\}$ or $\{2, 3\}$?
Is the lowest cost within label $\{1\}$ or $\{4\}$?

Weighted All Pairs (WAP): $\frac{K(K-1)}{2}$ binary classifiers (Beygelzimer, 2005)

is $\mathbf{c}[1]$ or $\mathbf{c}[4]$ lower?
—similar to CSOVO, with theoretically better way of calculating weights

Sensitive Error Correcting Output Code (SECOC): $(T \cdot K)$ binary classifiers (Langford, 2005)

is $\mathbf{c}[1] + \mathbf{c}[3] + \mathbf{c}[4]$ greater than some θ?

Extended Binary Classification: K binary classifiers (Lin, 2012)

is lowest-cost $y \leq$ some k?
—more proper for ordinal ranking

Outline

Key Idea: Cost Estimator

Goal

a classifier $g(\mathbf{x})$ that pays a small cost $\mathbf{c}[g(\mathbf{x})]$ on future **unseen** example $(\mathbf{x}, y, \mathbf{c})$

if every $\mathbf{c}[k]$ known

optimal

$$g^*(\mathbf{x}) = \text{argmin}_{1 \le k \le K}\, \mathbf{c}[k]$$

if $r_k(\mathbf{x}) \approx \mathbf{c}[k]$ well

approximately good

$$g_r(\mathbf{x}) = \text{argmin}_{1 \le k \le K}\, r_k(\mathbf{x})$$

how to get cost estimator r_k? **regression**

Cost Estimator by Per-class Regression

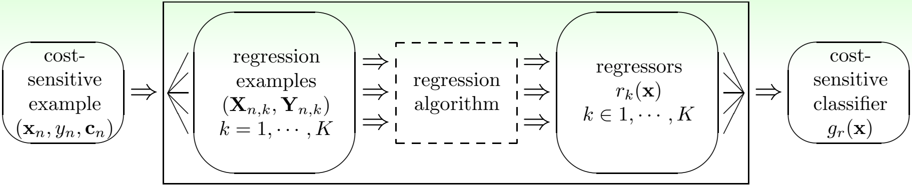
Given

N examples, each (input $\mathbf{x}_n$, label y_n, cost $\mathbf{c}_n$) $\in \mathcal{X} \times \{1, 2, \ldots, K\} \times R^K$

input	$\mathbf{c}_n[1]$	input	$\mathbf{c}_n[2]$	$\ldots$	input	$\mathbf{c}_n[K]$
$\mathbf{x}_1$	0,	$\mathbf{x}_1$	2,		$\mathbf{x}_1$	1
$\mathbf{x}_2$	1,	$\mathbf{x}_2$	3,		$\mathbf{x}_2$	5
$\ldots$						
$\mathbf{x}_N$	6,	$\mathbf{x}_N$	1,		$\mathbf{x}_N$	0
r_1		r_2			r_K	

want: $r_k(\mathbf{x}) \approx \mathbf{c}[k]$ for all future $(\mathbf{x}, y, \mathbf{c})$ and k

The Reduction Framework



1. transform cost-sensitive examples $(\mathbf{x}_n, y_n, \mathbf{c}_n)$ to regression examples $(\mathbf{x}_{n,k}, Y_{n,k}) = (\mathbf{x}_n, \mathbf{c}_n[k])$
2. use your favorite algorithm on the regression examples and get estimators $r_k(\mathbf{x})$
3. for each new input $\mathbf{x}$, predict its class using $g_r(\mathbf{x}) = \text{argmin}_{1 \le k \le K} r_k(\mathbf{x})$

the reduction-to-regression framework:

systematic & easy to implement

Theoretical Guarantees (1/2)

$$g_r(\mathbf{x}) = \operatorname*{argmin}_{1 \le k \le K} r_k(\mathbf{x})$$

Theorem (Absolute Loss Bound)

For any set of estimators (cost estimators) $\{r_k\}_{k=1}^{K}$ *and for any example* $(\mathbf{x}, y, \mathbf{c})$ *with* $\mathbf{c}[y] = 0$*,*

$$\mathbf{c}[g_r(\mathbf{x})] \le \sum_{k=1}^{K} \Big| r_k(\mathbf{x}) - \mathbf{c}[k] \Big|.$$

low-cost classifier $\Longleftarrow$ accurate estimator

Theoretical Guarantees (2/2)

$$g_r(\mathbf{x}) = \operatorname*{argmin}_{1 \le k \le K} r_k(\mathbf{x})$$

Theorem (Squared Loss Bound)

For any set of estimators (cost estimators) $\{r_k\}_{k=1}^{K}$ *and for any example* $(\mathbf{x}, y, \mathbf{c})$ *with* $\mathbf{c}[y] = 0$,

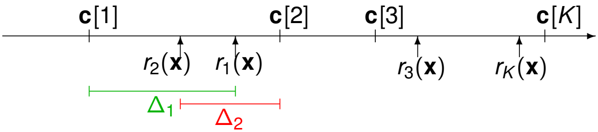
$$\mathbf{c}[g_r(\mathbf{x})] \le \sqrt{2 \sum_{k=1}^{K} \left(r_k(\mathbf{x}) - \mathbf{c}[k]\right)^2}.$$

applies to common **least-square regression**

A Pictorial Proof

$$\mathbf{c}[g_r(\mathbf{x})] \le \sum_{k=1}^{K} \left| r_k(\mathbf{x}) - \mathbf{c}[k] \right|$$

- assume $\mathbf{c}$ ordered and not degenerate:
  $y = 1;\ 0 = \mathbf{c}[1] < \mathbf{c}[2] \le \cdots \le \mathbf{c}[K]$
- assume mis-prediction $g_r(\mathbf{x}) = 2$:
  $r_2(\mathbf{x}) = \min_{1 \le k \le K} r_k(\mathbf{x}) \le r_1(\mathbf{x})$



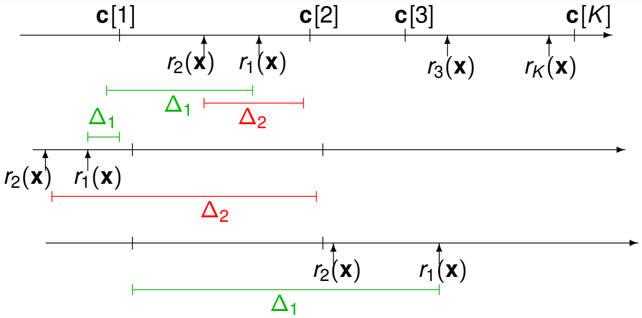
$$\mathbf{c}[2] - \underbrace{\mathbf{c}[1]}_{0} \le \left|\Delta_1\right| + \left|\Delta_2\right| \le \sum_{k=1}^{K} \left| r_k(\mathbf{x}) - \mathbf{c}[k] \right|$$

An Even Closer Look

let $\Delta_1 \equiv r_1(\mathbf{x}) - \mathbf{c}[1]$ and $\Delta_2 \equiv \mathbf{c}[2] - r_2(\mathbf{x})$

1. $\Delta_1 \geq 0$ and $\Delta_2 \geq 0$: $\mathbf{c}[2] \leq \Delta_1 + \Delta_2$
2. $\Delta_1 \leq 0$ and $\Delta_2 \geq 0$: $\mathbf{c}[2] \leq \Delta_2$
3. $\Delta_1 \geq 0$ and $\Delta_2 \leq 0$: $\mathbf{c}[2] \leq \Delta_1$

$$\mathbf{c}[2] \leq \max(\Delta_1, 0) + \max(\Delta_2, 0) \leq |\Delta_1| + |\Delta_2|$$



Tighter Bound with One-sided Loss

Define **one-sided loss** $\xi_k \equiv \max(\Delta_k, 0)$

with
$$\Delta_k \equiv \left(r_k(\mathbf{x}) - \mathbf{c}[k]\right) \text{ if } \mathbf{c}[k] = c_{\min}$$
$$\Delta_k \equiv \left(\mathbf{c}[k] - r_k(\mathbf{x})\right) \text{ if } \mathbf{c}[k] \neq c_{\min}$$

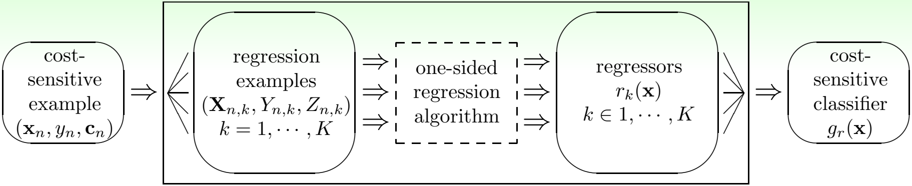
Intuition

- $\mathbf{c}[k] = c_{\min}$: wish to have $r_k(\mathbf{x}) \leq \mathbf{c}[k]$
- $\mathbf{c}[k] \neq c_{\min}$: wish to have $r_k(\mathbf{x}) \geq \mathbf{c}[k]$

—both wishes same as $\Delta_k \leq 0$ and hence $\xi_k = 0$

One-sided Loss Bound:
$$\mathbf{c}[g_r(\mathbf{x})] \leq \sum_{k=1}^{K} \xi_k \leq \sum_{k=1}^{K} \left|\Delta_k\right|$$

The Improved Reduction Framework



(Tu, 2010)

1. transform cost-sensitive examples $(\mathbf{x}_n, y_n, \mathbf{c}_n)$ to regression examples
2. use a one-sided regression algorithm to get estimators $r_k(\mathbf{x})$
3. for each new input $\mathbf{x}$, predict its class using $g_r(\mathbf{x}) = \text{argmin}_{1 \le k \le K} r_k(\mathbf{x})$

the reduction-to-OSR framework:
need a good OSR algorithm

Regularized One-sided Hyper-linear Regression

Given

$$\left(\mathbf{x}_{n,k}, Y_{n,k}, Z_{n,k}\right) = \left(\mathbf{x}_n, \mathbf{c}_n[k], 2\left[\!\left[\mathbf{c}_n[k] = \mathbf{c}_n[y_n]\right]\!\right] - 1\right)$$

Training Goal

all training $\xi_{n,k} = \max\left(\underbrace{Z_{n,k}\left(r_k(\mathbf{x}_{n,k}) - Y_{n,k}\right)}_{\Delta_{n,k}}, 0\right)$ small

—will drop k

$$\min_{\mathbf{w},b} \quad \frac{\lambda}{2}\langle \mathbf{w}, \mathbf{w}\rangle + \sum_{n=1}^{N} \xi_n$$

$$\text{to get} \quad r_k(\mathbf{x}) = \langle \mathbf{w}, \phi(\mathbf{x})\rangle + b$$

One-sided Support Vector Regression

Regularized One-sided Hyper-linear Regression

$$\min_{\mathbf{w},b} \quad \frac{\lambda}{2}\langle \mathbf{w}, \mathbf{w} \rangle + \sum_{n=1}^{N} \xi_n$$

$$\xi_n = \max\left(Z_n \cdot \left(r_k(\mathbf{x}_n) - Y_n\right), 0\right)$$

Standard Support Vector Regression

$$\min_{\mathbf{w},b} \quad \frac{1}{2C}\langle \mathbf{w}, \mathbf{w} \rangle + \sum_{n=1}^{N} (\xi_n + \xi_n^*)$$

$$\xi_n = \max\left(+1 \cdot \left(r_k(\mathbf{x}_n) - Y_n - \epsilon\right), 0\right)$$

$$\xi_n^* = \max\left(-1 \cdot \left(r_k(\mathbf{x}_n) - Y_n + \epsilon\right), 0\right)$$

OSR-SVM = SVR + $(0 \to \epsilon)$ + (keep ξ_n or ξ_n^* by Z_n)

OSR-SVM versus OVA-SVM

OSR-SVM: $g_r(\mathbf{x}) = \text{argmin}\ r_k(\mathbf{x})$

$$\min_{\mathbf{w},b} \quad \frac{\lambda}{2}\langle \mathbf{w}, \mathbf{w} \rangle + \sum_{n=1}^{N} \xi_n$$

with

$$r_k(\mathbf{x}) = \langle \mathbf{w}, \phi(\mathbf{x}) \rangle + b$$

$$\xi_n = \max\left(Z_n \cdot \left(r_k(\mathbf{x}_n) - Y_n\right), 0\right)$$

OVA-SVM: $g_r(\mathbf{x}) = \text{argmax}\ q_k(\mathbf{x})$

with

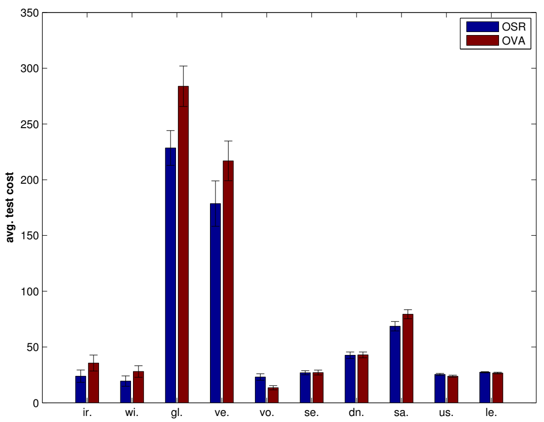
$$q_k(\mathbf{x}) = \langle \mathbf{w}, \phi(\mathbf{x}) \rangle + b$$

$$\xi_n = \max\left(-Z_n \cdot q_k(\mathbf{x}_n) + 1, 0\right)$$

OVA-SVM:

special case that replaces Y_n (i.e. $\mathbf{c}_n[k]$) by $-Z_n$

OSR-SVM on Semi-Real Data

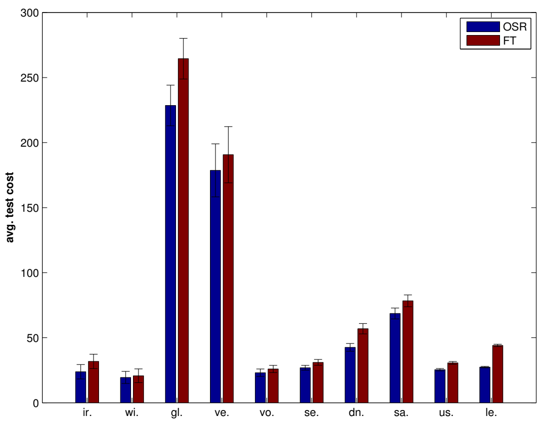


(Tu, 2010)

- OSR: a cost-sensitive extension of OVA
- OVA: regular SVM

OSR often significantly better than OVA

OSR versus FT on Semi-Real Data

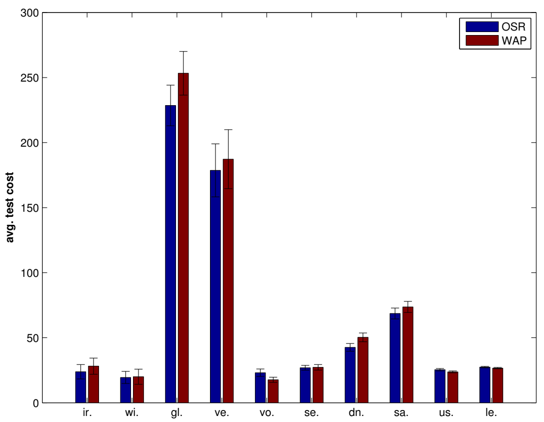


(Tu, 2010)

- OSR (per-class): $O(K)$ training, $O(K)$ prediction
- FT (tournament): $O(K)$ training, $O(\log_2 K)$ prediction

FT faster, but OSR better performing

OSR versus WAP on Semi-Real Data

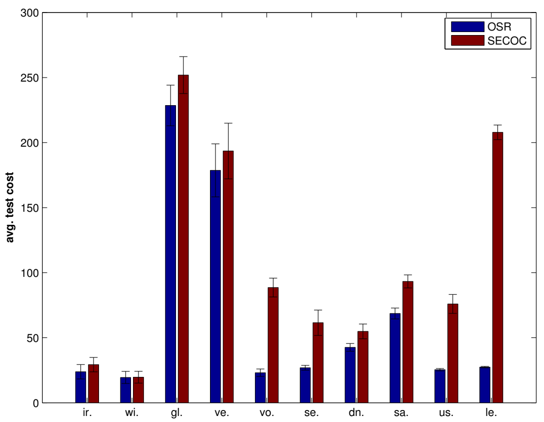


(Tu, 2010)

- OSR (per-class): $O(K)$ training, $O(K)$ prediction
- WAP (pairwise): $O(K^2)$ training, $O(K^2)$ prediction

OSR faster and comparable performance

OSR versus SECOC on Semi-Real Data

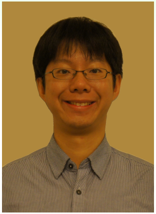


(Tu, 2010)

- OSR (per-class): $O(K)$ training, $O(K)$ prediction
- SECOC (error-correcting): big $O(K)$ training, big $O(K)$ prediction

OSR faster and much better performance

Biased Personal Favorites



- OSR: fast training, fast prediction, very good performance
- WAP or CSOVO: stable performance, pretty strong theoretical guarantee
- FT: fast training, very fast prediction, good performance, strong theoretical guarantee
- MetaCost if in the mood for Bayesian :-)

Outline

A Real Medical Application: Classifying Bacteria

The Problem

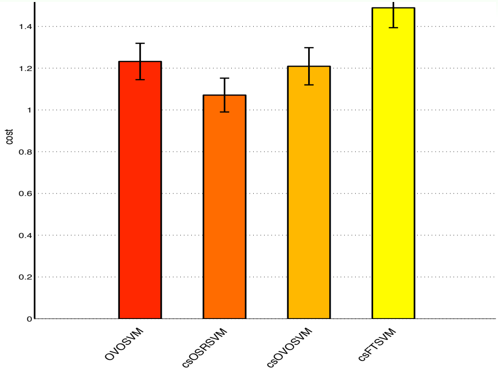
- by human doctors: **different treatments** $\Longleftrightarrow$ serious costs
- cost matrix averaged from two doctors:

	Ab	Ecoli	HI	KP	LM	Nm	Psa	Spn	Sa	GBS
Ab	0	1	10	7	9	9	5	8	9	1
Ecoli	3	0	10	8	10	10	5	10	10	2
HI	10	10	0	3	2	2	10	1	2	10
KP	7	7	3	0	4	4	6	3	3	8
LM	8	8	2	4	0	5	8	2	1	8
Nm	3	10	9	8	6	0	8	3	6	7
Psa	7	8	10	9	9	7	0	8	9	5
Spn	6	10	7	7	4	4	9	0	4	7
Sa	7	10	6	5	1	3	9	2	0	7
GBS	2	5	10	9	8	6	5	6	8	0

is cost-sensitive classification **realistic**?

OSR versus OVO/CSOVO/FT on Bacteria Data

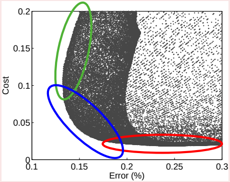
(Jan, 2011)



OSR best: cost-sensitive classification is helpful

Soft Cost-sensitive Classification

The Problem



- cost-sensitive classifier: low cost but high error
- traditional classifier: low error but high cost
- how can we get the blue classifiers?: low error and low cost

cost-and-error-sensitive: more suitable for medical needs

Improved OSR for Cost and Error on Semi-Real Data

key idea (Jan, 2012): consider a 'modified' cost that mixes original cost and 'regular cost'

Cost

iris	≈
wine	≈
glass	≈
vehicle	≈
vowel	◯
segment	◯
dna	◯
satimage	≈
usps	◯
zoo	◯
splice	≈
ecoli	≈
soybean	≈

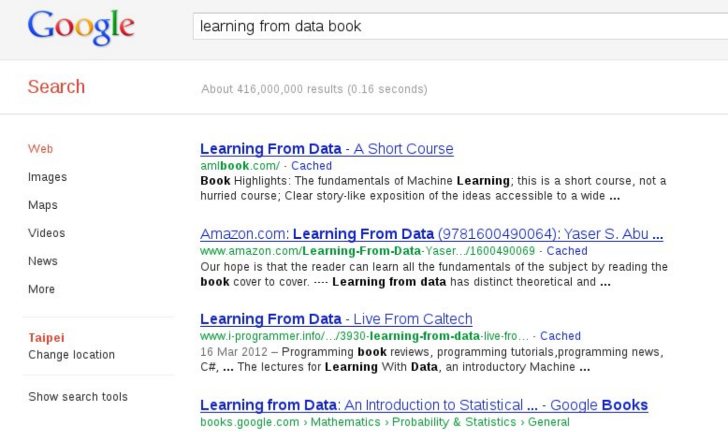
Error

iris	◯
wine	◯
glass	◯
vehicle	◯
vowel	◯
segment	◯
dna	◯
satimage	◯
usps	◯
zoo	◯
splice	◯
ecoli	◯
soybean	◯

improves other cost-sensitive classification algorithms, too

Outline

Preference Ranking in Search Engine



not just for searching **good machine learning book :-)**;
but also for **recommendation systems & other web service**

Three Properties of Search-Engine Ranking

- listwise with focus on **top ranks**
  - query-oriented & personalized
  - emphasis on **highly-preferred (relevant)** items
- **large scale**
  - both during **training** & testing
  - e.g. Yahoo! Learning-To-Rank Challenge 2010: 473K training URLs, 166K test URLs
- **ordinal data**
  - labeled qualitatively by human, e.g. { `highly irrelevant`, `irrelevant`, `neutral`, `relevant`, `highly relevant` }
  - **lack of quantitative info**

search-engine ranking problem:
learning a ranker from **large scale ordinal data**
with focus on **top ranks**

Search-Engine Ranking Setup

Given

for query indices $q = 1, 2, \cdots, Q$,

- a set of related documents $\{\mathbf{x}_{q,i}\}_{i=1}^{N(q)}$
- ordinal relevance $y_{q,i} \in \mathcal{Y} = \{0, 1, \ldots, K\}$ for each document $\mathbf{x}_{q,i}$

with large Q and $N(q)$

Goal

a ranker $r(\mathbf{x})$ that "accurately ranks" top $\mathbf{x}_{Q+1,i}$ from an **unseen** set of documents $\{\mathbf{x}_{Q+1,i}\}$

how to evaluate **accurate ranking around the top**?

Expected Reciprocal Rank (ERR; Chapelle, 2009)

assume for any example (document $\mathbf{x}$, rank y),

$$P(\text{user chooses document } \mathbf{x}) = (2^y - 1)/2^K$$

Assumption: Stopping Probability of **List of Documents**

$$\begin{aligned} & P(\text{user stops at position } i \text{ of list}) \\ = \; & P(\text{doesn't stop at pos. } i-1) \times P(\text{chooses document at pos. } i) \end{aligned}$$

ERR: Total Discounted Stopping Probability of List

$$ERR_q(r) \equiv \sum_{i=1}^{N(q)} \frac{1}{i} P(\text{user stops at position } i \text{ of the list ordered by } r)$$

large ERR $\Leftrightarrow$ small i matches large P
$\Leftrightarrow$ good ranking around top

Cost-Sensitive Ordinal Classification via Regression

Cost-Sensitive Ordinal Classification via Regression (COCR)

- reduction from listwise ranking (ERR) to cost-sensitive (ordinal) classification (approximately)
  —aim for **top rank** and **large scale data**
- reduction from cost-sensitive ordinal classification to binary classification
  —aim for **respecting ordinal data**
- reduction from binary classification to regression
  —aim for **large scale data** and **avoiding discrete ties**

costs can approximately embed true criteria of interest

Optimistic ERR (oERR) Cost for COCR

desired listwise criteria

How to make ERR(r) close to ERR(p), the ERR of perfect ranker?

embed criteria within cost

$$ERR(p) - ERR(r) \leq \blacksquare \cdot \left(\sum_{i=1}^{N(q)} \left(2^{y_{q,i}} - 2^{r(\mathbf{x}_{q,i})} \right)^2 + \Delta \right)$$

- $\Delta \approx 0$ if $r \approx p$ (optimistic)
- then, $\mathbf{c}[k] = \left(2^y - 2^k\right)^2$ embeds ERR
- oERR cost can then be coupled with other ordinal ranking techniques to improve performance

not a very tight bound, but **better than nothing**

COCR on Benchmark Data

(Ruan, 2013)

data set	Direct Regression	benchmark	oERR-COCR
LTRC1	0.4470	0.4484	0.4505
LTRC2	0.4440	0.4465	0.4461
MS10K	0.2643	0.2642	0.2792
MS30K	0.2748	0.2748	0.2942

- best ERR
- significantly better than direct regression

oERR-COCR **usually the best**

Outline

Summary

- **cost-sensitive binary classification**: just the **weights**
  - Bayesian: Approximate Bayes Optimal Decision (Elkan, 2001)
  - non-Bayesian: Cost-Proportionate Example Weighting (Zadrozny, 2003)
- **cost-sensitive binary classification**: **cost matrix/vectors**
  - Bayesian: MetaCost (Domingos, 1999)
  - non-Bayesian:
    Data Space Expansion (Abe, 2004) (to multiclass),
    Cost-Sensitive One-Versus-One (Lin, 2012), . . . (to binary),
    One-Sided Regression (Tu, 2010) (to regression)
    —most implemented here:
    `http://www.csie.ntu.edu.tw/~htlin/program/cssvm/`
- **beyond**:
  - cost-and-error-sensitive for medical application (Jan, 2012)
  - cost-sensitive, approximately, for information retrieval (Ruan, 2013)
  - cost-intervals (Liu, 2010)

discussion welcomed on algorithm and application opportunities

Giants' Shoulder

- binary:
  - Elkan, The Foundations of Cost-Sensitive Learning, 2001
  - Zadrozny et al., Cost-Sensitive Learning by Cost-Proportionate Example Weighting, 2003
  - Abu-Mostafa et al., Learning from Data: A Short Course, 2013
- multiclass:
  - Domingos, MetaCost: A General Method for Making Classifiers Cost-Sensitive, 1999
  - Abe et al., An Iterative Method for Multi-Class Cost-Sensitive Learning, 2004
  - Beygelzimer et al., Error Limiting Reductions Between Classification Tasks, 2005
  - Langford and Beygelzimer, Sensitive Error Correcting Output Codes, 2005
  - Beygelzimer et al., Multiclass Classification with Filter Trees, 2007
  - Chapelle et al., Expected Reciprocal Rank for Graded Relevance, 2009
  - Zhou and Liu, On Multi-class Cost-sensitive Learning, 2010.
  - Liu and Zhou, Learning with Cost Intervals, 2010.
  - Tu and Lin, One-Sided Support Vector Regression for Multiclass Cost-Sensitive Classification, 2010
  - Lin, A Simple Cost-Sensitive Multiclass Classification Algorithm Using One-Versus-One Comparisons, 2010
  - Jan et al., Cost-Sensitive Classification on Pathogen Species of Bacterial Meningitis by Surface Enhanced Raman Scattering, 2011
  - Lin and Li, Reduction from Cost-Sensitive Ordinal Ranking to Weighted Binary Classification, 2012
  - Jan et al., A Simple Methodology for Soft Cost-Sensitive Classification, 2012
  - Ruan et al., Improving Ranking Performance with Cost-Sensitive Ordinal Classification via Regression, 2013

Acknowledgments

- ACML Organizers!
- Computational Learning Lab @ NTU and Learning Systems Group @ Caltech for discussions

final advertisement :
my student's work on bipartite ranking (last talk of the conference)

Wei-Yuan Shen and Hsuan-Tien Lin. *Active Sampling of Pairs and Points for Large-scale Linear Bipartite Ranking*. ACML 2013.

Thank you. Questions?