Cost-sensitive Multiclass Classification via Regression

Hsuan-Tien Lin

Dept. of CSIE, NTU

Talk at NCU CSIE, 10/09/2013

Joint work with Han-Hsing Tu; parts appeared in ICML ’10
Which Digit Did You Write?

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

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

How can machines learn to classify?
Supervised Machine Learning

Cost-sensitive Classification

Parent

(picture, category) pairs

Kid's brain

good decision function

possibilities

Truth \( f(x) + \text{noise } e(x) \)

examples (picture \( x_n \), category \( y_n \))

learning algorithm

good decision function

learning model \( \{ g_\alpha(x) \} \)

challenge:

see only \( \{(x_n, y_n)\} \) without knowing \( f(x) \) or \( e(x) \)

\( \Rightarrow \) generalize to unseen \( (x, y) \) w.r.t. \( f(x) \)
Cost-sensitive Classification

Mis-prediction Costs \( g(x) \approx f(x) \) ?

- 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

Different applications:

**Evaluate mis-predictions differently**
ZIP Code Recognition

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

- **regular** classification problem: only right or wrong
- **wrong cost**: 1; **right cost**: 0
- prediction error of \( g \) on some \((x, y)\):

\[
\text{classification cost} = \left[ y \neq g(x) \right]
\]

**regular classification:**

well-studied, many good algorithms
Cost-sensitive Classification

Check Value Recognition

2

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

- **cost-sensitive** classification problem: different costs for different mis-predictions
- e.g. prediction error of $g$ on some $(x, y)$:

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

**cost-sensitive classification:**

**new**, need more research
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

<table>
<thead>
<tr>
<th>actual</th>
<th>predicted</th>
<th>H1N1</th>
<th>cold</th>
<th>healthy</th>
</tr>
</thead>
<tbody>
<tr>
<td>H1N1</td>
<td>0</td>
<td>1000</td>
<td>100000</td>
<td></td>
</tr>
<tr>
<td>cold</td>
<td>100</td>
<td>0</td>
<td>3000</td>
<td></td>
</tr>
<tr>
<td>healthy</td>
<td>100</td>
<td>30</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

- H1N1 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 terror but likes romance**

<table>
<thead>
<tr>
<th>actual</th>
<th>predicted</th>
<th>romance</th>
<th>fiction</th>
<th>terror</th>
</tr>
</thead>
<tbody>
<tr>
<td>romance</td>
<td>0</td>
<td>5</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>

**error measure = non-satisfaction**

**customer 2 who likes terror and romance**

<table>
<thead>
<tr>
<th>actual</th>
<th>predicted</th>
<th>romance</th>
<th>fiction</th>
<th>terror</th>
</tr>
</thead>
<tbody>
<tr>
<td>romance</td>
<td>0</td>
<td>5</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

**different customers:**

*evaluate mis-predictions differently*
### Cost-sensitive Classification Tasks

#### Movie Classification with Non-satisfaction

<table>
<thead>
<tr>
<th>Actual</th>
<th>Predicted</th>
<th>Romance</th>
<th>Fiction</th>
<th>Terror</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer 1, Romance</td>
<td>0</td>
<td>5</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>Customer 2, Romance</td>
<td>0</td>
<td>5</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

#### Patient Diagnosis with Society Cost

<table>
<thead>
<tr>
<th>Actual</th>
<th>Predicted</th>
<th>H1N1</th>
<th>Cold</th>
<th>Healthy</th>
</tr>
</thead>
<tbody>
<tr>
<td>H1N1</td>
<td>0</td>
<td>1000</td>
<td>100000</td>
<td></td>
</tr>
<tr>
<td>Cold</td>
<td>100</td>
<td>0</td>
<td>3000</td>
<td></td>
</tr>
<tr>
<td>Healthy</td>
<td>100</td>
<td>30</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

#### Check Digit Recognition with Absolute Cost

\[ C(y, g(x)) = |g(x) - y| \]
cost vector $\mathbf{c}$: a row of cost components

- customer 1 on a romance movie: $\mathbf{c} = (0, 5, 100)$
- an H1N1 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
Cost-sensitive Classification Setup

Given

\( N \) examples, each

\((\text{input } x_n, \text{label } y_n, \text{cost } c_n) \in X \times \{1, 2, \ldots, K\} \times R^K\)

- \( K = 2 \): binary; \( K > 2 \): multiclass
- will assume \( c_n[y_n] = 0 = \min_{1 \leq k \leq K} c_n[k] \)

Goal

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

- will assume \( c[y] = 0 = c_{\text{min}} = \min_{1 \leq k \leq K} c[k] \)
- note: \( y \) not really needed in evaluation

cost-sensitive classification:

  can express any finite-loss supervised learning tasks
Our Contribution

<table>
<thead>
<tr>
<th></th>
<th>binary</th>
<th>multiclass</th>
</tr>
</thead>
<tbody>
<tr>
<td>regular</td>
<td>well-studied</td>
<td>well-studied</td>
</tr>
<tr>
<td>cost-sensitive</td>
<td>known (Zadrozny, 2003)</td>
<td>ongoing (our work, among others)</td>
</tr>
</tbody>
</table>

A theoretic and algorithmic study of cost-sensitive classification, which ...

- introduces a methodology to reduce cost-sensitive classification to **regression**
- provides **strong theoretical support** for the methodology
- leads to a promising algorithm with **superior experimental results**

will describe the methodology and an algorithm
Key Idea: Cost Estimator

Goal

A classifier \( g(x) \) that pays a small cost \( c[g(x)] \) on future unseen example \((x, y, c)\)

- If every \( c[k] \) known:
  \[
  \text{optimal } g^*(x) = \arg\min_{1 \leq k \leq K} c[k]
  \]

- If \( r_k(x) \approx c[k] \) well:
  \[
  \text{approximately good } g_r(x) = \arg\min_{1 \leq k \leq K} r_k(x)
  \]

How to get cost estimator \( r_k \)? **Regression**
Cost Estimator by Per-class Regression

Given

\(N\) examples, each (input \(x_n\), label \(y_n\), cost \(c_n\)) \(\in \mathcal{X} \times \{1, 2, \ldots, K\} \times \mathbb{R}^K\)

<table>
<thead>
<tr>
<th>input</th>
<th>(c_n[1])</th>
<th>input</th>
<th>(c_n[2])</th>
<th>\ldots</th>
<th>input</th>
<th>(c_n[K])</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x_1)</td>
<td>0,</td>
<td>(x_1)</td>
<td>2,</td>
<td></td>
<td>(x_1)</td>
<td>1</td>
</tr>
<tr>
<td>(x_2)</td>
<td>1,</td>
<td>(x_2)</td>
<td>3,</td>
<td></td>
<td>(x_2)</td>
<td>5</td>
</tr>
<tr>
<td>(\ldots)</td>
<td>(\ldots)</td>
<td>(\ldots)</td>
<td>(\ldots)</td>
<td>(\ldots)</td>
<td>(\ldots)</td>
<td>(\ldots)</td>
</tr>
<tr>
<td>(x_N)</td>
<td>6,</td>
<td>(x_N)</td>
<td>1,</td>
<td></td>
<td>(x_N)</td>
<td>0</td>
</tr>
</tbody>
</table>

\[r_1\]

\[r_2\]

\[r_K\]

want: \(r_k(x) \approx c[k]\) for all future \((x, y, c)\) and \(k\)
The Reduction Framework

1. Transform cost-sensitive examples \((x_n, y_n, c_n)\) to regression examples \((X_{n,k}, Y_{n,k})\) where \(k = 1, \ldots, K\).
2. Use your favorite algorithm on the regression examples and get regressors \(r_k(x)\).
3. For each new input \(x\), predict its class using \(g_r(x) = \arg\min_{1 \leq k \leq K} r_k(x)\).

The reduction-to-regression framework:

systematic & easy to implement
Theoretical Guarantees (1/2)

\[ g_r(x) = \arg\min_{1 \leq k \leq K} r_k(x) \]

**Theorem (Absolute Loss Bound)**

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

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

low-cost classifier \(\iff\) accurate regressor
Theoretical Guarantees (2/2)

\[ g_r(x) = \arg\min_{1 \leq k \leq K} r_k(x) \]

Theorem (Squared Loss Bound)

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

\[ c[g_r(x)] \leq \sqrt{2 \sum_{k=1}^{K} (r_k(x) - c[k])^2}. \]

applies to common least-square regression
A Pictorial Proof

\[ c[gr(x)] \leq \sum_{k=1}^{K} |r_k(x) - c[k]| \]

- Assume \( c \) ordered and not degenerate:
  \( y = 1; 0 = c[1] < c[2] \leq \cdots \leq c[K] \)
- Assume mis-prediction \( gr(x) = 2: \)
  \( r_2(x) = \min_{1 \leq k \leq K} r_k(x) \leq r_1(x) \)

\[ c[2] - c[1] \leq |\Delta_1| + |\Delta_2| \leq \sum_{k=1}^{K} |r_k(x) - c[k]| \]
An Even Closer Look

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

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

$c[2] \leq \max(\Delta_1, 0) + \max(\Delta_2, 0) \leq |\Delta_1| + |\Delta_2|$
Tighter Bound with One-sided Loss

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

with

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

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

**Intuition**

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

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

**One-sided Loss Bound:**

\[ c[g_r(x)] \leq \sum_{k=1}^{K} \xi_k \leq \sum_{k=1}^{K} |\Delta_k| \]
Reduction to Regression

The Improved Reduction Framework

1. Transform cost-sensitive examples \((x_n, y_n, c_n)\) to regression examples \((X_{n,k}, Y_{n,k}, Z_{n,k})\) for each \(k = 1, \ldots, K\):
   \[
   (X_{n,k}, Y_{n,k}, Z_{n,k}) = (x_n, c_n[k], 2[c_n[k] = c_n[y_n]] - 1)
   \]

2. Use a one-sided regression algorithm to get regressors \(r_k(x)\).

3. For each new input \(x\), predict its class using \(g_r(x) = \arg\min_{1 \leq k \leq K} r_k(x)\).

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

\[(X_{n,k}, Y_{n,k}, Z_{n,k}) = (x_n, c_n[k], 2[c_n[k] = c_n[y_n]] - 1)\]

Training Goal

all training \(\xi_{n,k} = \max \left( \frac{Z_{n,k} (r_k(X_{n,k}) - Y_{n,k})}{\Delta_{n,k}}, 0 \right)\) small

—will drop \(k\)

\[\min_{w,b} \frac{\lambda}{2} \langle w, w \rangle + \sum_{n=1}^{N} \xi_{n}\]

to get \(r_k(X) = \langle w, \phi(X) \rangle + b\)
One-sided Support Vector Regression

Regularized One-sided Hyper-linear Regression

\[
\min_{w,b} \quad \frac{\lambda}{2} \langle w, w \rangle + \sum_{n=1}^{N} \xi_n \\
\xi_n = \max \left( Z_n \cdot (r_k(X_n) - Y_n) , 0 \right)
\]

Standard Support Vector Regression

\[
\min_{w,b} \quad \frac{1}{2C} \langle w, w \rangle + \sum_{n=1}^{N} (\xi_n + \xi_n^*) \\
\xi_n = \max \left( +1 \cdot (r_k(X_n) - Y_n - \epsilon) , 0 \right) \\
\xi_n^* = \max \left( -1 \cdot (r_k(X_n) - Y_n + \epsilon) , 0 \right)
\]

\[\text{OSR-SVM} = \text{SVR} + (0 \rightarrow \epsilon) + (\text{keep } \xi_n \text{ or } \xi_n^* \text{ by } Z_n)\]
Comparisons

OSR versus Other Reductions

**OSR: \( K \) regressors**

How unlikely (costly) does the example belong to class \( k \)?

**Filter Tree (FT): \( K - 1 \) binary classifiers**

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

**Weighted All Pairs (WAP): \( \frac{K(K-1)}{2} \) binary classifiers**

Is \( c[1] \) or \( c[4] \) lower?

**Sensitive Error Correcting Output Code (SECOC): \( (T \cdot K) \) bin. cla.**

Experiment: OSR-SVM versus OVA-SVM

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

OSR often significantly better than OVA
Comparisons

Experiment: OSR versus FT

FT faster, but OSR better performed

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

Experiment: OSR versus WAP

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

OSR faster and comparable performance
Comparisons

Experiment: OSR versus SECOC

- 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
Conclusion

- **reduction to regression**: a simple way of designing cost-sensitive classification algorithms
- **theoretical guarantee**: absolute, squared and **one-sided** bounds
- **algorithmic use**: a **novel and simple** algorithm OSR-SVM
- **experimental performance of OSR-SVM**: **leading** in SVM family

more algorithm and **application** opportunities
Acknowledgments

- Profs. Chih-Jen Lin, Yuh-Jyh Lee, Shou-de Lin for suggestions
- Prof. Chia-Hui Chang for talk invitation
- Computational Learning Lab @ NTU for discussions

Thank you. Questions?