A Comparison of Methods for Cost-sensitive Support Vector Machines

TeKang Jan

Adviser: Dr. Hsuan-Tien Lin

Computational Learning Lab

August 2, 2010
What is cost-sensitive classification?

Cost-sensitive classification
regular classification with cost information
What kinds of credit cards should be assigned?

Credit card business

- Credit card business in the UK is estimated US$750 million.
- Assigning the **appropriate credit limit** to the applications is more and more important.
- Category: Unacceptable, Student card, Platinum card

Table: cost of the bank

<table>
<thead>
<tr>
<th>appropriate status</th>
<th>Unacceptable</th>
<th>Student card</th>
<th>Platinum card</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unacceptable</td>
<td>0</td>
<td>20000</td>
<td>1000000</td>
</tr>
<tr>
<td>Student card</td>
<td>2000</td>
<td>0</td>
<td>3000</td>
</tr>
<tr>
<td>Platinum card</td>
<td>100000</td>
<td>10000</td>
<td>0</td>
</tr>
</tbody>
</table>
example: a credit card application

- an application apply to HIGH credit limit

<table>
<thead>
<tr>
<th>appropriate status</th>
<th>classify to</th>
<th>Unacceptable</th>
<th>Student card</th>
<th>Platinum card</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unacceptable</td>
<td>0</td>
<td>20000</td>
<td>1000000</td>
<td></td>
</tr>
<tr>
<td>Student card</td>
<td>2000</td>
<td>0</td>
<td>3000</td>
<td></td>
</tr>
<tr>
<td>Platinum card</td>
<td>100000</td>
<td>10000</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

a cost-sensitive example becomes

- (Application 1, Unacceptable, [0, 20000, 1000000])

cost vector: use label and cost matrix to generate
Set up

an unknown distribution \( D \) on \( \mathcal{X} \times \mathcal{Y} \times \mathbb{R}^K \)

where \( \mathcal{X} \subseteq \mathbb{R}^d \), \( \mathcal{Y} = \{1, 2, \ldots, K\} \)

- cost-sensitive example = \( (x, y, c) \in \mathcal{X} \times \mathcal{Y} \times \mathbb{R}^K \)
- cost-sensitive training set \( S_c = \{(x_n, y_n, c_n)\}_{n=1}^N \)

we assume \( c_n[y] = 0 \), \( \forall n = 1, \ldots, N \)

Goal

learn a classifier \( g: \mathcal{X} \rightarrow \mathcal{Y} \) which minimizes

\[
E_{x, y, c \sim D}[c[g(x)]]
\]
Support vector machine

SVM was designed for binary classification.

“Large margin” and “Kernel mapping” make SVM powerful
Multi-class SVM

SVM solves binary classification problems

How to extend it for multi-class?

One group of methods
combine several SVMs

Other group of methods
all-together approach
Regular multi-class SVM: one-versus-one

Figure: each SVM is trained for two classes

\[ \binom{K}{2} \] SVMs, predict one class with the largest vote
Cost-sensitive multi-class SVM: csOVOSVM

Figure: each SVM is trained for two classes

csOVOSVM
relabel cost-sensitive examples to weighted examples
Regular multi-class SVM: filter tree

Figure: which label set has the desired label?

hold a single elimination tournament on the set of labels
Cost-sensitive multi-class SVM: csFTSVM

Figure: which label set has the lowest cost?

csFTSVM
relabel cost-sensitive examples to weighted examples
Regular multi-class SVM: one-versus-all

Figure: each SVM predicts the similarity score for corresponding class

SVM with the **highest output** function predicts the class
Cost-sensitive multi-class SVM: csOSRSVM

Figure: each SVM predicts the estimated cost for corresponding class

**csOSRSVM**

SVM with the *lowest output* function predicts the class.
All-together methods

an approach for multi-class SVM problems by solving a single optimization problem
All-together methods

Cost-sensitive: Crammer and Singer
Contributions

- design a novel cost-sensitive classification of all-together formulation algorithm
- derive theoretical guarantee
- compare performance with the cost-sensitive methods above
Crammer and Singer formulation

**Goal**

learn a classifier $g: \mathcal{X} \rightarrow \mathcal{Y}$ which minimizes test error

like OVA formulation: each SVM predicts the similarity score for corresponding class

**Framework**

$g(x) = \text{argmax}_{k=1...K} \mathbf{w}_k^T x$: find $K$ vectors $\mathbf{w}_k$, $\forall k = 1 \ldots K$

**Similarity score**

pick label $y$ that has the highest score

How to get good vectors $\mathbf{w}_k$, $\forall k = 1 \ldots K$?
One-versus-all formulation

\[
\min_{\mathbf{w}_k, \xi_n} \quad \frac{1}{2} \sum_{k=1}^{K} \mathbf{w}_k^T \mathbf{w}_k + C \sum_{n=1}^{N} \xi_n^k \\
\text{subject to} \quad \mathbf{w}_k^T \mathbf{x}_n \geq 1 - \xi_n^k \quad \text{if } y_n = k \\
\mathbf{w}_k^T \mathbf{x}_n \leq -1 + \xi_n^k \quad \text{if } y_n \neq k \\
\xi_n^i \geq 0, \quad n = 1, \ldots, N
\]
Crammer and Singer formulation

\[
\begin{align*}
\min_{w_k, \xi_n} & \quad \frac{1}{2} \sum_{k=1}^{K} w_k^T w_k + C \sum_{n=1}^{N} \xi_n \\
\text{subject to} & \quad w_{y_n}^T x_n - w_k^T x_n \geq e_n^k - \xi_n, \quad n = 1, \ldots, N \\
\text{where} & \quad e_n^k = \begin{cases} 
0 & \text{if } y_n = k \\
1 & \text{if } y_n \neq k
\end{cases}
\end{align*}
\]
Graph illustration of Crammer and Singer formulation

**Figure:** left figure: classify correctly, middle figure: classify correctly but suffer some loss right figure: classify wrong

\[
\begin{align*}
\min_{w_k, \xi_n} & \quad \frac{1}{2} \sum_{k=1}^{K} w_k^T w_k + C \sum_{n=1}^{N} \xi_n \\
\text{subject to} & \quad w_{y_n}^T x_n - w_k^T x_n \geq e_n^k - \xi_n, n = 1, \ldots, N
\end{align*}
\]
How to extend Crammer and Singer formulation?

All-together methods

Cost-insensitive: Crammer and Singer

\[
\min_{w_k, \xi_n} \frac{1}{2} \sum_{k=1}^{K} w_k^T w_k + C \sum_{n=1}^{N} \xi_n
\]

subject to \( w_{y_n}^T x_n - w_k^T x_n \geq e_n^k - \xi_n, \quad n = 1, \ldots, N \)

Cost-sensitive: Crammer and Singer

How to add cost information reasonably?
Multi-class Cost-sensitive classification
Multi-class cost-sensitive SVM
Experiments

How to add cost information reasonably?

Regression

in OSR, they use regressions to predict costs

OSR

\[
\begin{align*}
\min_{w_k, \xi_n} & \quad \frac{1}{2} \sum_{k=1}^{K} w_k^T w_k + C \sum_{n=1}^{N} \xi_n^k \\
\text{subject to} & \quad w_k^T x_n \leq c_n[y_n] + \xi_n^k \quad \text{if} \quad y_n = k \\
& \quad w_k^T x_n \geq c_n[k] - \xi_n^k \quad \text{if} \quad y_n \neq k
\end{align*}
\]

(1) - (2) \rightarrow w_k^T x_n - w_{y_n}^T x_n \geq (c_n[k] - c_n[y_n]) - \xi_n'}
How to add cost information reasonably?

by this observing, we can change Crammer and Singer formulation to cost sensitive Crammer and Singer formulation

$$\min_{w_k, \xi_n} \frac{1}{2} \sum_{k=1}^{K} w_k^T w_k + C \sum_{n=1}^{N} \xi_n$$

subject to

$$w_{y_n}^T x_n - w_k^T x_n \geq e_k^n - \xi_n, \quad n = 1, \ldots, N$$

where

$$e_k^n = \begin{cases} 
0 & \text{if } y_n = k \\
1 & \text{if } y_n \neq k 
\end{cases}$$

replace $e_k^n$ to $c_n[k]$
Cost-sensitive Crammer and Singer formulation

\[
\begin{align*}
\min_{w_k, \xi_n} & \quad \frac{1}{2} \sum_{k=1}^{K} w_k^T w_k + C \sum_{n=1}^{N} \xi_n \\
\text{subject to} & \quad w_{y_n}^T x_n - w_k^T x_n \geq c_n[k] - \xi_n, \quad n = 1, \ldots, N
\end{align*}
\]

Decision function

\[
\arg\max_{k=1, \ldots, K} w_k^T x
\]
The advantage of Cost-sensitive Crammer and Singer

- good bound of this formulation
The advantage of Cost-sensitive Crammer and Singer

take a step back to OSR

\[ \{r_k(x)\}_{k=1}^K \] are regressions for predicting cost \( c[k], k = 1, \ldots K \)

**General regression loss (quadratic loss)**

\[
\text{loss} = \sum_{k=1}^{K} (r_k(x) - c[k])^2
\]

**Special regression loss (one sided loss)**

\[
\text{loss} = \sum_{k=1}^{K} \max((2[I(c[k] = c_{\min}] - 1)(r_k(x) - c[k]), 0))
\]
Cost-sensitive Crammer and Singer bound

Special regression loss (one sided loss)

\[
\text{loss} = \sum_{k=1}^{K} \max\left(\left(2 \left[ c[k] = c_{\min} \right] - 1 \right) \left( r_k(x) - c[k] \right), 0 \right)
\]

define \( \Delta_y = r_y(x) - c[y] \) and \( \Delta_k = c[k] - r_k(x), \forall k \neq y \)

Theorem (per-example loss bound)

for any cost-sensitive example \((x, y, c)\)

\[
c[g(x)] \leq \max(\Delta_y + \Delta_g(x), 0) \leq \max_k \max(\Delta_y + \Delta_k, 0)
\]
Optimization step of cost-sensitive Crammer and Singer

- We don’t give the optimization step here.
- Details can be found in my thesis.
Experiments

Data set
- eight benchmark data sets from UCI repository
- randomly separate, 75% for training, 25% for testing, run for 20 times

Cost generation
- cost vector from a cost matrix $C$
- $C(y, y) = 0$
- $C(y, k) = \text{uniformly sampled from } \left[0, 100\frac{|\{n:y_n=k\}|}{|\{n:y_n=y\}|}\right], y \neq k$
Data statistics

**Table:** problems statistics

<table>
<thead>
<tr>
<th>data set</th>
<th># examples</th>
<th># class</th>
<th># features</th>
</tr>
</thead>
<tbody>
<tr>
<td>iris</td>
<td>150</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>wine</td>
<td>178</td>
<td>3</td>
<td>13</td>
</tr>
<tr>
<td>glass</td>
<td>214</td>
<td>6</td>
<td>9</td>
</tr>
<tr>
<td>vehicle</td>
<td>846</td>
<td>4</td>
<td>18</td>
</tr>
<tr>
<td>vowel</td>
<td>990</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>segment</td>
<td>2310</td>
<td>7</td>
<td>19</td>
</tr>
<tr>
<td>dna</td>
<td>3186</td>
<td>3</td>
<td>180</td>
</tr>
<tr>
<td>satimage</td>
<td>6435</td>
<td>6</td>
<td>36</td>
</tr>
</tbody>
</table>
Is Cost-sensitive Crammer and Singer learning from cost?

csCSSVM can really learn from the cost information
Is Cost-sensitive Crammer and Singer good?

csCSSVM performance is better than csFTSVM
Is Cost-sensitive Crammer and Singer good?

csCSSVM, csOSRSVM and csOVOSVM results are similar
Why their result are similar?

- csCSSVM and csOSRSVM formulations are somehow similar
- Are the bounds they optimize also similar?
Compare csCSSVM and csOSRSVM

The bound of csOSRSVM

\[ c[g_{osr}(x)] \leq \sum_{k=1}^{K} \max(\Delta_k, 0) \]

bound1

The bound of csCSSVM

\[ c[g_{cs}(x)] \leq \max_k \max(\Delta_y + \Delta_k, 0) \]

bound2
Compare their bounds using csOSRSVM bound (bound1)
Compare their bounds using csCSSVM bound (bound2)
Multi-class Cost-sensitive classification
Multi-class cost-sensitive SVM
Experiments

Number of unique support vectors

<table>
<thead>
<tr>
<th>Method</th>
<th>Iris</th>
<th>Wine</th>
<th>Glass</th>
<th>Vehicle</th>
<th>Vowel</th>
<th>Segment</th>
<th>DNA</th>
<th>Satimage</th>
</tr>
</thead>
<tbody>
<tr>
<td>csFTSVM</td>
<td>0</td>
<td>500</td>
<td>1000</td>
<td>1500</td>
<td>2000</td>
<td>2500</td>
<td></td>
<td></td>
</tr>
<tr>
<td>csOVOSVM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CSSVM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>csOSRSVM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>csCSSVM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

csCSSVM, csOSRSVM can use fewer unique support vectors
Conclusion

- csCSSVM can really learn from the cost information
- csCSSVM, csOSRSVM, and csOVOSVM results are similar
- csFTSVM and CSSVM is worse than csCSSVM
- csCSSVM, csOSRSVM can use fewer unique support vectors

To summarize, csCSSVM is a good multi-class cost-sensitive algorithm

but not better than csOSRSVM

Any question?