Some Aspects of Classification: Imbalance & Multi-Class cases

Evgeny Burnaev

Skoltech, Moscow, Russia

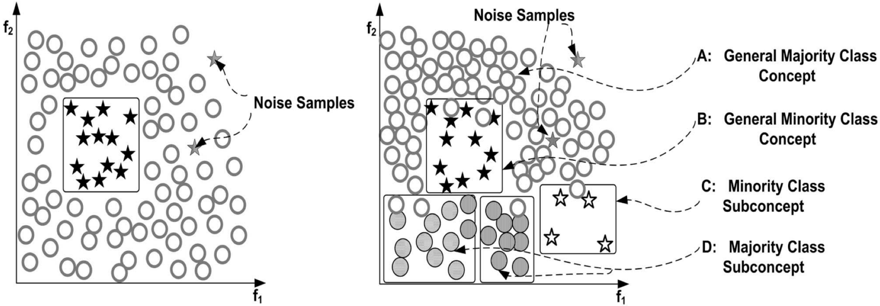
Outline

1. Imbalanced Classification

2. Multi-Class Classification

Introduction

- Between-class imbalance (relative imbalance)
- Relative imbalance vs. imbalance due to rare instances or "absolute rarity"
    - — Within class imbalance
- Data complexity vs. imbalanced data vs. small sample size



Introduction

- Binary classification: often dataset has unavoidable (natural) imbalance
- *Minor class* (of **prime** interest) vs. *major class*: e.g. classification of "cancerous" vs. "healthy" mammography image
- Standard classifiers (SVM, kNN, log. reg., etc.): classes are equally important $\Rightarrow$ results are biased towards the major class
- Poor prediction of minor class while the average quality can be good:
  - — target events occurs in 1% of all cases,
  - — classifier always gives a 'no-event' answer,
  - — it is wrong just 1% of all cases

- Approaches to increase importance of the minor class:
  - Adapt a probability threshold for classifiers,
  - Modify a loss function, e.g., by assigning more weight to the minor class error,
  - Resample a dataset in order to soften or remove class imbalance
- We focus on resampling: convenient, allows to use standard classifiers
- The main aim:
  - review and compare main resampling methods,
  - compare strategies of resampling amount (i.e., how many observations to add or drop) selection,
  - explore their influence on quality of classification

Notations and Problem Statement

- Dataset $S_m = (\mathbf{x}_i, y_i)_{i=1}^m$, where $\mathbf{x}_i \in \mathbb{R}^N$, $y_i \in \{0, 1\}$
- $C_0(S_m) = \{(\mathbf{x}_i, y_i) \in S_m \mid y_i = 0\}$ is a major class,
- $C_1(S_m) = \{(\mathbf{x}_i, y_i) \in S_m \mid y_i = 1\}$ is a minor class, i.e. $|C_0(S_m)| > |C_1(S_m)|$
- *Imbalance ratio* $IR(S_m) = \frac{|C_0(S_m)|}{|C_1(S_m)|}$, $IR(S_m) \geq 1$

Learning a classifier

- Llearn a classifier using imbalanced training sample S_m,
- The dataset S_m is *resampled* using a method r:
  - — some observations in S_m are dropped, or
  - — some new synthetic observations are added to S_m
- The result of resampling is a dataset $r(S_m)$ with $IR(r(S_m)) < IR(S_m)$,
- Standard classification model h is learned on $r(S_m)$ to construct a classifier $h_{r(S_m)} : \mathbb{R}^N \to \{0, 1\}$

- Performance is determined by a predefined *classifier quality metrics* $Q(h_{S_{train}}, S_{test})$ (e.g. AUC under Precision-Recall curve):
  - — input classifier $h_{S_{train}}$,
  - — testing dataset S_{test},
  - — the higher value is the better
- k-fold cross-validation is used to estimate $Q^{CV}(S_m)$

Overview of Resampling Methods

Resampling method r:

1. Takes input:
   - dataset S_m;
   - *resampling multiplier* $m > 1$ for resulting imbalance ratio $IR(r(S_m)) = \frac{1}{m} \cdot IR(S_m)$;
   - additional parameters, specific for the method
2. Add synthesized objects to the minor class (*oversampling*), or drop objects from the major class (*undersampling*), or both
3. Outputs resampled dataset $r(S_m)$ with imbalance ratio $IR(r(S_m)) = \frac{1}{m} \cdot IR(S_m)$

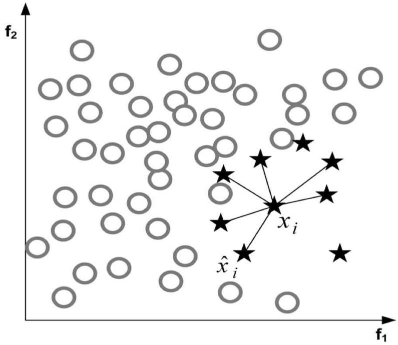
Random Oversampling (ROS)

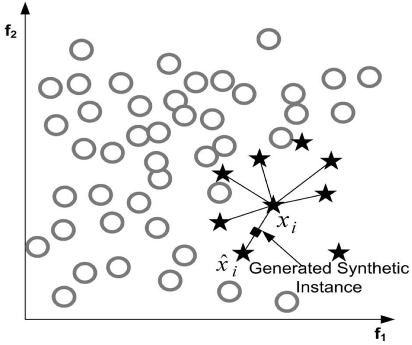
- ROS, also known as bootstrap oversampling
- No additional input parameters
- It adds to the minor class new $(m-1)|C_1(S_m)|$ objects
- Each of objects is drawn from uniform distribution on $C_1(S_m)$

Random Undersampling (RUS)

- No additional input parameters
- It chooses random subset of $C_0(S_m)$ with $|C_0(S_m)|\frac{m-1}{m}$ elements and drops it from the dataset
- All subsets of $C_0(S_m)$ have equal probabilities to be chosen

Synthetic Minority Oversampling Technique (SMOTE)





Synthetic Minority Oversampling Technique (SMOTE)

- Input parameter: k (number of neighbors)
- Oversampling, it adds to the minor class new synthesized objects
- Initialize: $S_{new} := \emptyset$. Repeat $(m-1)|C_1(S_m)|$ times:
  1. Select randomly $x_i \in C_1(S_m)$
  2. Find k minor class NN of x_i, randomly select x_j from them
  3. Select randomly x on the segment connecting x_i and x_j
  4. $S_{new} := S_{new} \cup \{(x, 1)\}$
- Add objects to the dataset: $\tilde{S} = S_m \cup S_{new}$

Artificial Data

- Artificial pool of data with ~ 1000 datasets
- Artificial datasets were drawn from a Gaussian mixture distribution
- Each of two classes is represented as a Gaussian mixture with not more than 3 components
- Number of features varies from 6 to 40, size of dataset from 200 to 1000, IR from 0.05 to 0.35.

Real Data

- Real pool of data with ~ 100 datasets
- Different areas: biology, medicine, engineering, sociology
- All features are numeric or binary, their number varies from 3 to 1000
- Size of dataset varies from 200 to 1000, IR from 0.02 to 0.75

Setup of Experiments

- For each dataset we varied classifier model, resampling method and resampling multiplier
- We used Bootstrap, RUS and SMOTE with $k = 5$
- We varied resampling multiplier from 1.25 to 10.0
- We used Decision Trees, k-Nearest Neighbors, and Logistic Regression with ℓ_1 regularization
- Optimal parameters of a classifier were selected by cross-validation

Setup of Experiments

- Accuracy measure = Area under precision-recall curve Q_{PRC}
- We performed 10-fold cross-validation and calculated Q_{PRC}^{CV} — average of Q_{PRC}

Resampling Multiplier Selection

- Two strategies of resampling multiplier selection:
  - — equalizing strategy, *EqS*: select multiplier providing balanced classes ($IR = 1$) in resulting dataset
  - — CV-search, *CVS*: select optimal multiplier (i.e., providing maximum of Q^{CV}) by cross-validation
- The equalizing strategy seems to be reasonable as it removes class imbalance which we initially tried to tackle. It is quick and widely used
- CV-search may provide better quality but it is more time-consuming

Dolan-More Curves

- $\{r_1, \ldots, r_n\}$ — the set of considered resampling methods
- $\{S_1, \ldots S_T\}$ — the set of tasks (datasets),
- q_{ti} — the quality of the method i on the dataset t,
- $p_i(\beta)$ is a fraction of datasets, on which the method i is worse than the best one not more than β times:

$$p_i(\beta) = \frac{1}{T}\left|\left\{t : q_{ti} \geq \frac{1}{\beta}\max_i q_{ti}\right\}\right|, \ \beta \geq 1$$

Dolan-More Curves

- $p_i(1)$ is a fraction of datasets where the method i is the best
- A graph of $p_i(\beta)$ on β is called Dolan-More curve for the method i
- The higher the curve, the better the method!

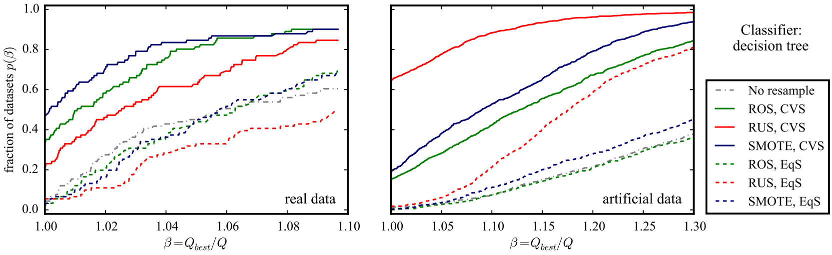


Figure : Dolan-More curves for metric Q_{PRC}^{CV}

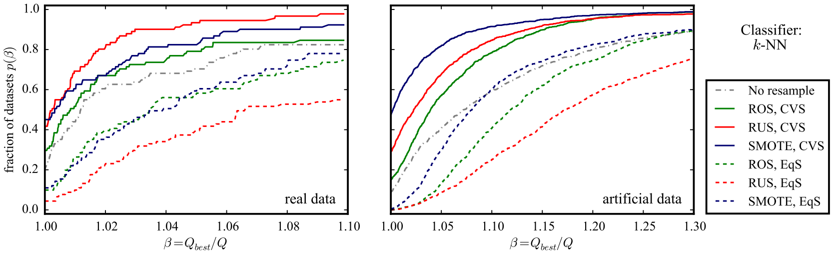


Figure : Dolan-More curves for metric Q_{PRC}^{CV}

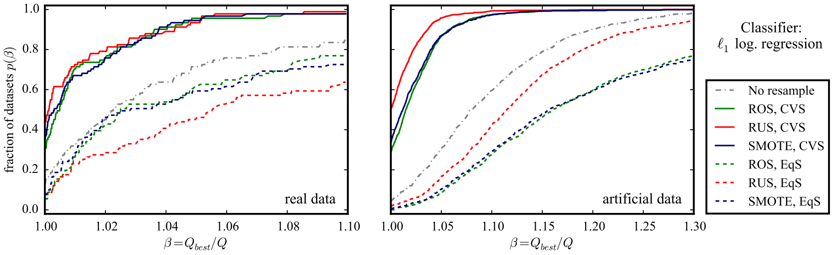


Figure : Dolan-More curves for metric Q_{PRC}^{CV}

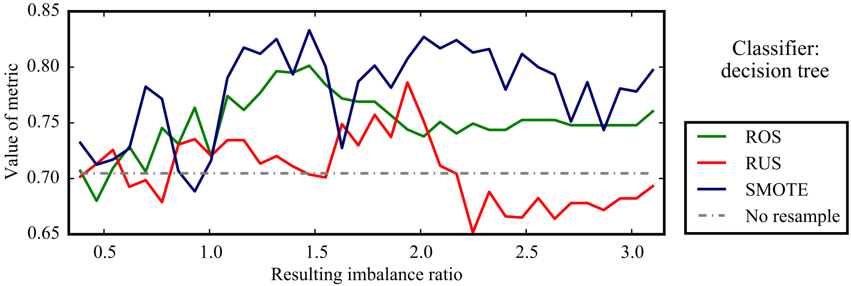


Figure : Value of Q^{CV}_{PRC} vs. resulting IR for dataset "Delft pump 1x3"

Conclusions

- Influence of resampling on the quality strongly depends on resampling multiplier
- All resampling methods with CV-search of multiplier improve the quality on most datasets, especially for Decision trees and Logistic regression
- The equalizing strategy of multiplier selection (EqS) shows much lower quality, and it is even worse than no resampling for k-Nearest neighbors and Logistic regression

Conclusions

- Performance of resampling method depends on the classifier used, and there is no method that would always outperform the others
- Impact of resampling on quality depends on the data it is applied to. E.g. RUS EqS used with Decision tree demonstrates this distinctly: it is worse than no resampling for the real datasets but outperforms it on the artificial data
- Classification without resampling is the best choice in some cases. E.g., for Logistic regression it is about 15% of real datasets and 5% of artificial

Conclusions

- The overall conclusion is the following. Resampling improves classification of imbalanced datasets in most cases if a method and a multiplier are selected properly. But if not, resampling may have negative effect on quality of classification
- So, to improve quality of classification, one has to determine optimal resampling method (also considering no resampling) and multiplier in every particular imbalanced task

Multi-Class Classification Problem

- **Training sample**: i.i.d. generated by D

$$S_m = \{(\mathbf{x}_i, y_i)\}_{i=1}^m \in X^m \times Y^m$$

  — mono-label case: $\mathrm{Card}(Y) = K$
  — multi-label case: $Y = \{-1, +1\}^K$
- **Problem**: find classifier $h : X \to Y$ in H with small generalization error
  — mono-label case: $R_D(h) = \mathbb{E}_{\mathbf{x}\sim D}\left[1_{h(\mathbf{x})\neq f(\mathbf{x})}\right]$
  — multi-label case:
  $R_D(h) = \mathbb{E}_{\mathbf{x}\sim D}\left[\frac{1}{K}\sum_{j=1}^K 1_{[h(\mathbf{x})]_j \neq [f(\mathbf{x})]_j}\right]$

Comments

- Usually $K \leq 100$
- If $K \gg 1$ then some other methods are used, e.g. ranking
- Big values of K increases computational burden
- In general, classes are not balanced

One-vs-All

- **Technique**
  - — for each class $k \in Y$ learn a binary classifier

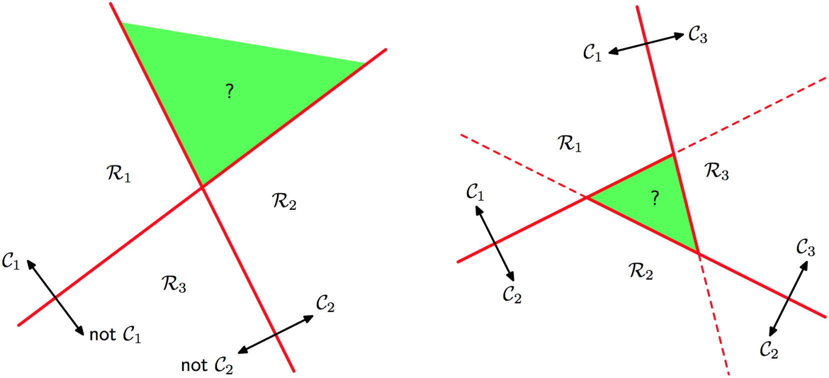
$$h_k(\mathbf{x}) = \mathrm{sign}(f_k(\mathbf{x}))$$

  - — combine binary classifiers via voting, e.g. majority voting

$$h:\ \mathbf{x} \to \arg\max_{k \in Y} f_k(\mathbf{x})$$

- **Comments**
  - — calibration: classifiers scores are not comparable
  - — simple and frequently used in practice, computational advantages in some cases

One-vs-All



Consider the use of $K - 1$ classifiers each of which solves a two-class problem of separating points in a particular class from points not in that class. This approach leads to regions of input space that are ambiguously classified

One-vs-One

- **Technique**
  - for each pair $(k, k') \in Y$, $k \neq k'$ learn a binary classifier $h_{k,k'} : X \to \{0, 1\}$
  - combine binary classifiers via majority vote

$$h(\mathbf{x}) = \arg\max_{k' \in Y} |\{k : h_{k,k'}(\mathbf{x}) = 1\}|$$

- **Comments**
  - computational complexity: train $K(K-1)/2$ binary classifiers
  - overfitting: size of a training sample can be small for a given pair of classifiers

Approach based on Error-Correcting Codes

- 8 classes, codes of length 6

codes

classes	1	2	3	4	5	6
1	0	0	0	1	0	0
2	1	0	0	0	0	0
3	0	1	1	0	1	0
4	1	1	0	0	0	0
5	1	1	0	0	1	0
6	0	0	1	1	0	1
7	0	0	1	0	0	0
8	0	1	0	1	0	0

$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$	$f_5(x)$	$f_6(x)$
0	1	1	0	1	1

new example x

Approach based on Error-Correcting Codes

- Assign L-long binary code word to each class, i.e. represent each class as

$$\mathbb{C} = [\mathbb{C}_{k,j}] \in \{0, 1\}^{[1,K]\times[1,L]}$$

- Learn a binary classifier $f_j : X \to \{0, 1\}$ for each column. Example $\mathbf{x}$ in class k is labeled with $\mathbb{C}_{k,j}$
- Classifier output:

$$\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_L(\mathbf{x})),$$

- Final classifier

$$h : \mathbf{x} \to \arg\min_{k\in Y} d_{\text{Hamming}}\left(\mathbb{C}_{k,\cdot}, \mathbf{f}(\mathbf{x})\right)$$

Comments

- One-vs-all approach is the most widely used
- No clear empirical evidence of the superiority of other approaches
- Large structured multi-class problems are often treated as ranking problems
- Above we considered how to reduce multi-class classification to the binary case. Also we can incorporate a multi-class structure explicitly into a classification algorithm, see e.g. multi-class logistic regression or multi-class SVM (below)

Multi-Class SVMs

- **Optimization problem**

$$\min_{\mathbf{w},\xi} \frac{1}{2} \sum_{k=1}^{K} \|\mathbf{w}_k\|^2 + C \sum_{i=1}^{m} {}_i$$

$$s.t.\ \mathbf{w}_{y_i}^{\mathrm{T}} \mathbf{x}_i + \delta_{y_i,k} \geq \mathbf{w}_k^{\mathrm{T}} \mathbf{x}_i + 1 - \xi_i$$

$$(i,k) \in [1,m] \times Y$$

- **Decision function**:

$$h:\ \mathbf{x} \to \arg\max_{k \in Y} (\mathbf{w}_k^{\mathrm{T}} \mathbf{x}) = \arg\max_{k \in Y} \left(\sum_{i=1}^{m} \alpha_{i,k} (\mathbf{x}_i \cdot \mathbf{x}) \right),$$

  where $\{\alpha_{i,k}\}_{i=1}^{m}$, $k \in Y$ are dual variables
- Complex constraints, $m \cdot K$ size