Upsampling, a comparative study with new ideas

Joseph R. Barr
Machine Intelligence Lab
Acronis SCS
Scottsdale, Arizona, USA
joe.barr@acronisscs.com

Marcus Sobel
Department of Statistics
Temple University, Philadelphia
Pennsylvania, USA
marcus.sobel@temple.edu

Tyler Thatcher
Machine Intelligence Lab
Acronis SCS, Scottsdale
Arizona, USA
tyler.thatcher@acronisSCS.com

***Abstract*—We give a brief tour of sampling techniques not uncommon in dealing with imbalanced data having binary tags. We mention the rather 'naive' upsampling techniques, classical bootstrap (B. Efron, 1979) and a more recent technique called SMOTE (Chawla, et al, 2002). We demonstrate those techniques using features derived from the Android OS source code with tags derived from the COMMON VULNERABILITIES & EXPOSURES public database (CVE, Mitre) which contains known software bugs in widely-used applications. Evidently, the data which contains approximately 0.5% positive tags with remaining approximately 99.5% negative tags is highly, in fact extremely imbalanced. A new idea is to synthesize samples from a minority class using an idea from the Bayesian domain. Forthcoming is an empirical analysis of the procedure.**

Keywords: Unstructured Data, Android Source Code Imbalanced Data, Binary Tags, Binary Classification, ROC, AUC, Lift, Specificity, Recall, Upsampling, Bootstrapping, SMOTE, Bayesian Upsampling, EM, Autoregressive

1. Introduction

It's not uncommon, in fact, quite common that labeled data which is available to the statistician is imbalanced, sometime grossly imbalanced. Faced with a classification task, imbalanced binary data poses a significant challenge. For example, if one is solely concerned with 'match-rate', then for data having 99% tagged as -1 and 1% tagged as $+1$, a 'silly' classifier which blindly predicts all records as -1 will invariably have a 99% match rate. However, there's little debate as to the utility (or the wisdom) of such classifier. After all, the main reason for developing a classifier, especially for imbalanced data, is to be able to effectively cull out the minority class. (Model) generalization is a standard adjudicator of a classifier: whether a classifier generalizes on holdout datasets. Furthermore, and as a guiding light of model efficacy, in the presence the BIAS-VARIANCE TRADEOFF, (Figure 1), it's imperative to control model complexity. Model's goodness is measured against ROC, gains/lift, sensitivity and specificity.

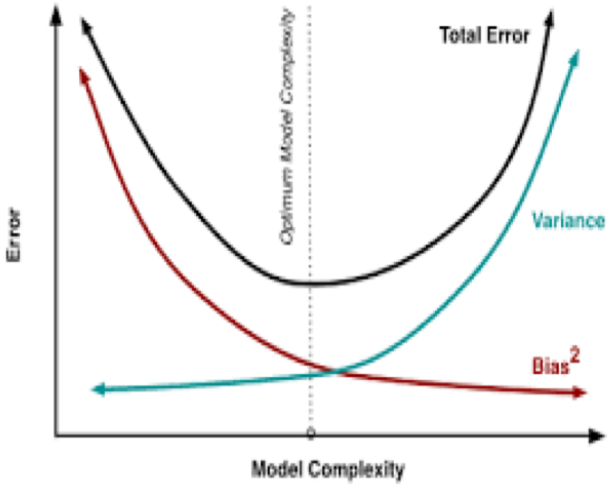

Figure 1: Bias-variance tradeoff.

2. The Data

We demonstrate the efficacy of various sampling techniques using the Android OS source code. The tags are derived from MITRE's database called COMMON VULNERABILITIES & EXPOSURES.

For more information on Mitre, see https://www.mitre.org. And for CVE, see https://cve.mitre.org/cve/. A summary of the data is displayed in the tables below.

C & C++ code	**Totals**
Lines of code	39,353,232
Number of files	103,111
Size (approx. in MB)	1,864
Number of functions	1,154,416
Tokens count	204,777,237
Unique tokens count	3,964,717
Number of CVEs	6,775

Table 1: C and C++ Android source code

Using auto-encoding with LSTM network, functions (more

precisely, the tokens representing functions) are mapped into a 128D Euclidean space. We append the auto-encoder with four additional CODE SMELL-inspired features
1. The size of function or number of tokens making up a function.
2. The number of parameter passed into the function.
3. The length of the longest line.
4. Density of tokens shared with amalgam of CVE's.

In total an input dimensionality is $128 + 4 = 132$ ($\mathbb{R}^{132}$), and a binary output (+1 for CVE and 0 for non-CVE.)
There are 6,775 CVEs and 1,154,416 functions. With less than 0.6 percent tagged as $+1$, the data is highly imbalanced.

2.1. Dealing with imbalanced data

As we saw above, the data $\mathcal{D} \subset \mathcal{X} \times \mathcal{Y} \subset \mathbb{R}^{132} \times \{-1, +1\}$ consists of over 95.5% with label -1 and less than 0.5% labeled $+1$. This is a problematic state of affairs if one wishes to construct an effective classifier. There are several possible approaches to deal with imbalanced data, all involve UPSAMPLING of one kind or another. We investigate, compare and contrast three upsampling methods using standard goodness-of-fit metrics.

3. Feature engineering & embedding functions into a Euclidean space

The ENCODER-DECODER paradigm with an AUTO-ENCODER [3] is put to work to embed functions into a Euclidean space $\mathbb{R}^d$. A vector $\mathcal{V}_{\mathcal{F}} \in \mathbb{R}^d$ representation of a function (or method) $\mathcal{F}$ generated by an auto-encoder are used as *features* of a classifier. There are several viable options to embed functions. We've opted for BYTE-PAIR ENCODING and an *Long Short-Term Memory (LSTM)*. To overcome the OUT-OF-VOCABULARY problem, we've implemented a workflow which incorporate chopping tokens into *sub-tokens* pairs of bytes, using an algorithm called BYTE-PAIR ENCODING and then embedding those sub-tokens using *Long Short-Term Memory (LSTM)* network. Finally we re-assemble the original token by aggregating the sub-tokens (e.g., average corresponding vectors.) Additional features are 1) LENGTH OF FUNCTION, number of tokens in function/method; 2) LENGTH OF LONGEST LINE; 3) NUMBER OF PARAMETERS; 4) NUMBER OF 'BAD WORDS' SHARED WITH CVEs.

We demonstrate the approach with $d = 128$, i.e., functions are embedded into $\mathbb{R}^{128}$, plus the four features described above. Together with the binary tags, the training pairs are (x, y), with $x \in \mathbb{R}^{132}$ and $y \in \{-1, +1\}$ with $y = +1$ stands in for CVE and $y = -1$ for non-CVE.

4. Upsampling

A rather naive approach, which may very well be a good choice is to upsample minority class. A plausible approach would sample a substantial portion of minority class, say 80 percent and will sample an equal number of examples from the majority class. This valid technique works well when the minority class is sufficient substantial, say higher than 2% in million records. In which case we may sample examples 15,000 in that minority class and an equal number of records of a majority class; this leaves 5,000 positive records and mixed with an equal number of negative examples, yields a validation set of 10,000 on which to test a classifier. ROC. Visual in Figure 2.

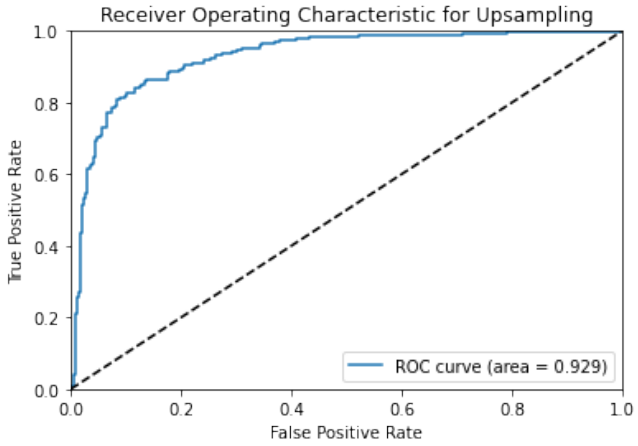

Figure 2: ROC curve with upsampling.

'

5. Bootstrapping

Bootstrapping [2], developed by Bradley Efron in the late 70s has become very popular variance-reduction technique. The idea is that sampling with replacement, although not independent, under mild conditions bootstrap converge (in some technical sense. This is known as 'mixing'). Thanks to Leo Breiman keen eye, Bootstrapping has begat 'bagging' which has begat 'random forests'. 1994, 1996 and 2006, respectively. We bootstrap a hundred (100) random samples (with replacement) and calculate AUC, specificity and recall. Figure 3 is a graphic depiction of the results

6. SMOTE: Synthetic Minority Over-sampling Technique

With over 15,000 citations, SMOTE, Chawla, Bowyer, et al. 2002, [1], a technique which simulates minority tags has been very popular to amplify faint signal due to dearth of examples of a minority class. SMOTE synthesises examples by simulating examples similar to the minority class. With an adjustable parameter Python's implementations of SMOTE

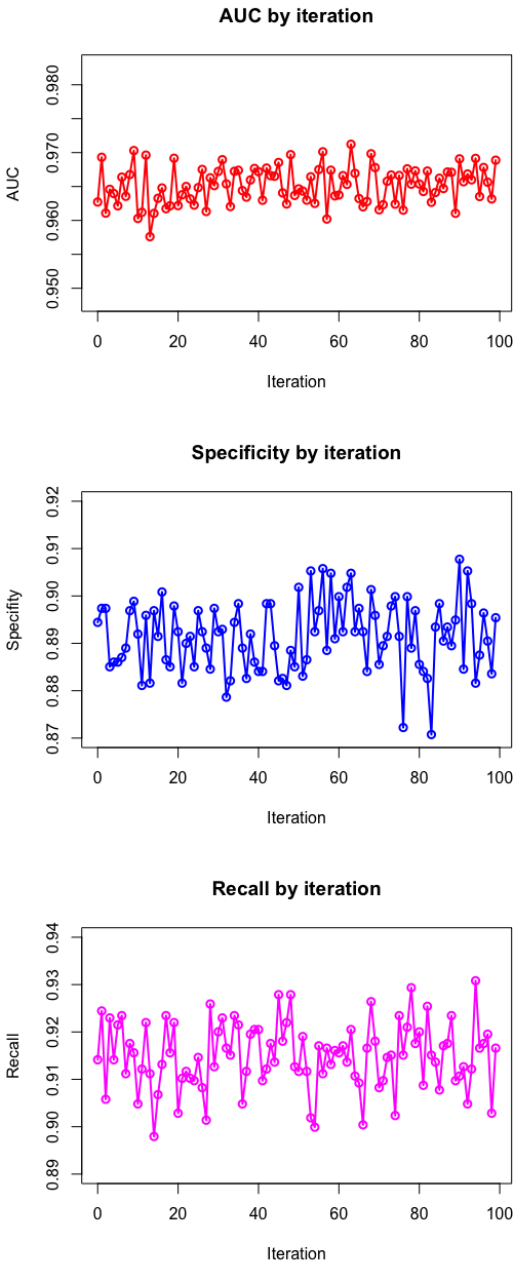

Figure 3: Bootstrap. Fluctuation of metrics by iteration.

allows for simulating minority class at various levels. We demonstrate performance with various levels of synthetic examples. ROC, AUC and Gains Chart/Lift with SMOTE at various levels is given in Figures 4 and 5.

7. Bayesian 'SMOTE', an optimal over-sampling technique

We describe a recursive Bayesian method to simulate minority samples. As always, at our disposal are

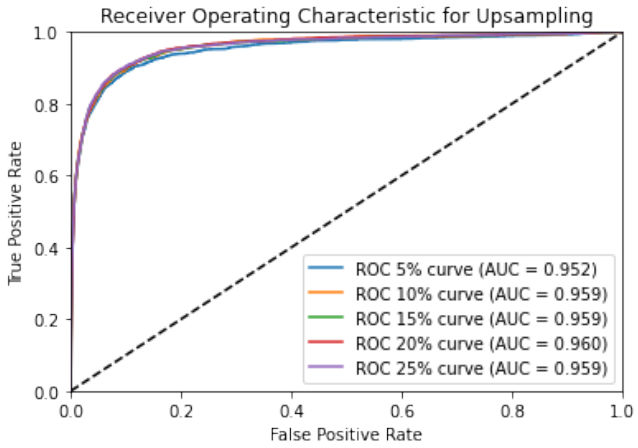

Figure 4: SMOTE: AUC at various levels.

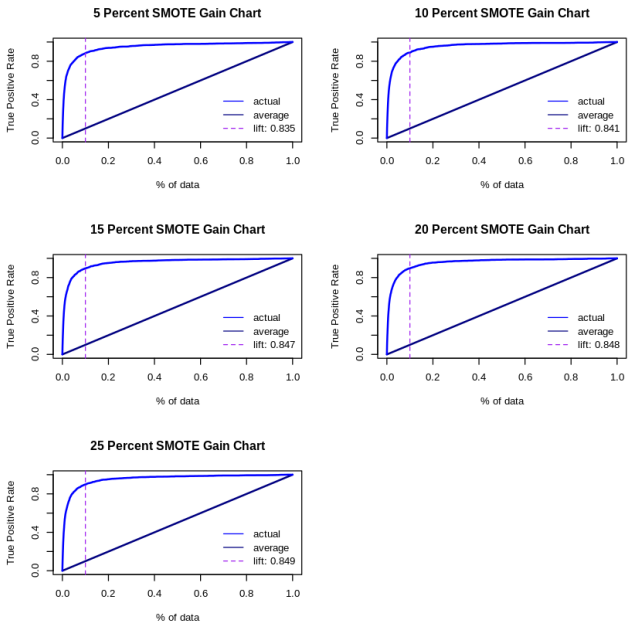

Figure 5: SMOTE: Gains & lift at various levels.

training pairs $\mathcal{D}$ consisting of (x, y) with $x \in \mathbb{R}^d$, and $y \in \{-1, +1\}$. The idea is to start with the minority class, say $P^1 = \{(x, y) \in \mathcal{D} : y = +1\}$ and augment it by simulating new samples.

Introduce a new parameter $0 \leq \alpha \leq 1$ which distinguishes the convex combinations between simulated vectors and their nearest neighbors.

Base step.
Select X_1 using a density estimate for the minority class. Select α_1 uniformly from the unit interval.

Stage s step

At stage s we have already constructed minority covariate vectors:

$$S = \{X_1(\alpha_1), ..., X_s(\alpha_s)\}$$

taking the form,

$$X_j(\alpha_j) = \sigma(\alpha_j)X_j^* + (1 - \sigma(\alpha_j))X_{j,NN}^*; \qquad j = 1, ..., s$$

($\sigma(\bullet)$ denotes the sigmoid function)

New Step

Select a single covariate vector called X_{s+1}^* from the minority class and select its nearest neighbor (from among the remaining minority vectors) $X_{s+1,NN}^*$ Define the convex combination:

$$X_{s+1}^*(\alpha_{s+1}) = \sigma(\alpha_{s+1})X_{s+1}^* + (1 - \sigma(\alpha_{s+1}))X_{s+1,NN}^*$$

Auto-regressive assumption
We assume the α's, defined above, are auto-regressively generated. For the sequence $\alpha_1, \alpha_2, ...\alpha_s$, with hyperparameters, $\lambda, \omega, \tau > 0$, we assume that:

$$\alpha_{s+1} = \lambda\alpha_s + \omega + \epsilon_{s+1}, \quad \epsilon_{s+1} \sim N(0, \tau^2). \qquad (1)$$

Recursive step.

For $s > 1$, D the dimensionality of the minority data, κ a small number, ϕ the standard multivariate normal density, and assuming $\alpha_1, ..., \alpha_s$ have been estimated, compute the density, $\widehat{f}(x|\alpha_{s+1})$ of X_{s+1} by

$$\widehat{f}(x|\alpha_{s+1}) = \frac{1}{s}\sum_{j=1}^{s} \phi\left(\frac{x - \sigma(\alpha_j)X_j - (1 - \sigma(\alpha_j))X_{j,NN}}{\kappa}\right) \frac{1}{(2\pi\kappa^2)^{D/2}} \qquad (2)$$

with

$$\sigma(\alpha_1)X_1 + (1-\sigma(\alpha_1))X_{1,NN}, ..., \sigma(\alpha_s)X_s + (1-\sigma(\alpha_s))X_{s,NN}$$

being those points already selected. This is the Parzen density estimator for the sample already selected.

Use the EM algorithm described below to estimate the parameters α, λ, ω and τ. Step (1) is the E step while step (2) corresponds to the M step.

EM Algorithm

1) At stage s having estimated $\alpha_1, ..., \alpha_s$ and having estimated λ, ω, and τ, we estimate α_{s+1} by

$$\alpha_{s+1} = \arg\min_{\alpha_{s+1}} \left(\widehat{f}(X_{s+1}|\alpha_{s+1}) \exp\big(\frac{-(\alpha_{s+1} - \lambda\alpha_s - \omega)^2}{2\tau^2}\big)/\tau\right) \qquad (3)$$

2) Estimate λ, ω and τ by

$$\lambda, \omega, \tau = \arg\max_{\lambda,\omega,\tau} \Big(\log \widehat{f}(X_{s+1}|\alpha_{s+1}) - \sum_j \frac{(\alpha_{j+1} - \lambda\alpha_j)^2}{2\tau^2} - s\log(\tau)\Big). \qquad (4)$$

3) Continue until convergence.

8. Conclusions

To date, 2021, SMOTE and its cousin ADASYN: Adaptive Synthetic Sampling Method for Imbalanced Data [4] is a state of the art procedure to effectively classify two-class dataset with imbalanced data. Given the flurry of activity in this particular area of statistics & machine learning, one must ask themselves why there are few, if any alternatives to SMOTE and whether a better, more effective procedure to simulate examples of a minority class may exist. We would like to think that the answer is in the affirmative.

9. Bibliography

[1] **SMOTE: Synthetic Minority Over-sampling Technique**, N.V. Chawla, K. W. Bowyer, L.O. Hall, W.P. Kegelmeyer, Journal of Artificial Intelligence Research 16 (2002) 321–357

[2] **Bootstrap Methods: Another Look at the Jackknife**, B. Efron, Ann. Statist. 7(1): 1-26 (January, 1979)

[3] **Deep Learning**, I. Goodfellow, Y. Bengio and A. Courville, MIT Press, (2016)

[4] **ADASYN: Adaptive Synthetic Sampling Approach for Imbalanced Learning**, Haibo He, Yang Bai, Edwardo A. Garcia, and Shutao Li, (2008)

[5] **Particle Learning for General Mixtures**, Carlos M. Carvalho, Hedibert F. Lopes, Nicholas G. Polson and Matt A. Taddy, Bayesian Analysis (2010)