2016 IEEE 16th International Conference on Data Mining Workshops

# Sacrificing overall classification quality to improve classification accuracy of well-sought classes

Kevin M. Amaral
kevin.m.amaral@gmail.com
Department of Computer Science
University of Massachusetts Boston

Ping Chen
Ping.Chen@umb.edu
Department of Engineering
University of Massachusetts Boston

Wei Ding
ding@cs.umb.edu
Department of Computer Science
University of Massachusetts Boston

Rajani Sadasivam
Rajani.Sadasivam@umassmed.edu
Division of Health Informatics and Implementation Science
University of Massachusetts Medical School

***Abstract*—Classification has been an active field in machine learning for decades. With many methods proposed for various topics in classification, this paper intends to show some initial ideas and findings in one classification scenario where accuracy of only one or a few classes is greatly valued, while the other classes are not important. Using a neural network model and challenging real world dataset, our preliminary results showed the accuracy of important class was significantly improved by sacrificing the accuracy of unimportant classes.**

## I. Introduction

Generally, the goal of classification is to identify each class with a high rate of accuracy [2]. However, there are cases in which we are willing to sacrifice the accuracy of less valuable classes to improve the accuracy of a well-sought subset of classes especially in real-world data. In this paper, we will describe a real-world scenario in which we want to maximize the accuracy of an individual class in an unbalanced dataset with little regard for those separated out. Further, we will show the success of creating sacrificial additional classes and the circumstances in which they out-perform a binary classification.

## II. Method

We adopt a run-of-the-mill neural network as our classification model, which includes one input layer, two hidden layers, one logistic regression layer, and one Softmax + Argmax Layer for single class output [1]. The detailed process is described below.

### A. Formal Construction

First, instances were organized in an array $X$ as column vectors. Let $W_1$ be the weight matrix between the input layer and the first hidden layer and let $\vec{b}_1$ be the bias term corresponding to the first hidden layer. Let $\tanh$ represent the component-wise hyperbolic tangent function. $O_1$ is the matrix of output vectors from the first hidden layer.

Let $W_2$ be the weight matrix between the output of the first hidden layer and second hidden layer and let $\vec{b}_2$ be the bias term corresponding to the second hidden layer. $O_2$ is the matrix of output vectors from the second hidden layer. This has the same overall structure as the previous layer.

Let $W_3$ be the weight matrix between the second hidden layer and the logistic regression layer and let $\vec{b}_3$ be the bias term corresponding to the logistic regression layer. Let $\sigma_i$ be the $i$th component of the softmax function over appropriately-many dimensions, the number of classes. Let $\mathcal{C}$ be the collection of classes as numeric indexes. $\hat{y}$ is the row-vector corresponding to the predicted class labels of each data instance in $X$.

Our model is formally calculated as

$$O_1 = \tanh\left(W_1 X + \vec{b}_1\right) \tag{1}$$

$$O_2 = \tanh\left(W_2 O_1 + \vec{b}_2\right) \tag{2}$$

$$\hat{y} = \underset{c \in \mathcal{C}}{\arg\max} \quad \sigma_c\left(W_3 O_2 + \vec{b}_3\right) \tag{3}$$

We define the addition above between matrices and column vectors $\vec{b}_i$ is defined as the addition of $\vec{b}_i$ to each column of the matrix.

In training the model, mean negative log likelihood was used as our measure of fitness. Below is the formula for mean negative log likelihood:

$$\widehat{\mathrm{NLL}} = -\mathbb{E}_c\left(\log\left(\sigma_c\left(W_3 O_2 + \vec{b}_3\right)\right)\right) \tag{4}$$

Minimizing the negative log likelihood function is equivalent to maximizing the likelihood function by the monotonicity property of the natural logarithm. Though they take their extrema at the same points, the log of the likelihood function has nicer properties.

Likelihood values are more often than not strictly less than 1. Taking derivatives of the likelihood function with respect to a large number of parameters will strain the precision of system in which the model was built and run. Taking the log alleviates the precision strain by mapping values less than 1 to larger negative values with larger differences.

2375-9259/16 $31.00 © 2016 IEEE
DOI 10.1109/ICDMW.2016.190

Many more niceties of the log likelihood function are realized with respect to actually calculating the derivatives. In our work, we utilized the auto-differentiation system included in Theano [6] so as not to preoccupy ourselves with the difficulty of calculating differentials regardless of the ease awarded by our choice of fitness measure.

Each of our weight matrices $W_i$ and bias terms $\vec{b}_i$ were updated during the training process using gradient descent with $\lambda = 0.3$. The update rules are shown below:

$$w_i^{j,k} \leftarrow w_i^{j,k} - \lambda \frac{\partial \widehat{\text{NLL}}}{\partial w_i^{j,k}} \tag{5}$$

$$b_i^{j} \leftarrow b_i^{j} - \lambda \frac{\partial \widehat{\text{NLL}}}{\partial b_i^{j}} \tag{6}$$

## III. Evaluation With A Real-world Application

### A. Dataset Description

We evaluated our approach with a real-world Share2Quit dataset [4], which consists of smokers who were invited into the Share2Quit program. In this program, smokers were encouraged and incentivized to participate in an online peer recruitment study. The goal of this study was for smokers to recruit their friends and family smokers to also participate in an online tobacco cessation intervention [5]. This recruitment portion is usually the focus of smoking-cessation researchers.

Each user was provided an opportunity to recruit friend and family smokers for 30 days. A successful recruitment was defined as when a peer recruited smoker registered on the online tobacco intervention. Further details of this study has been published in [3]. Afterwards, each member they recruited would count towards their individual recruitment counts. Each recruiter was incentivized up until their seventh (7th) recruit, after which they did not receive additional rewards for recruitment. As can be later seen in our numbers, recruiters with recruitment counts of seven (7) spike, after which very few continue to recruit more members.

The overall goal is to identify and target specifically the users who are willing to recruit many additional members to a program given incentives. If such members can be identified, fiscal resources can be pooled and directed towards those members to maximize the program's reach within a limited budget, without wasting funds on ineffective members.

Our dataset consists of a total of 1643 users, 264 of which actively recruited and 1379 of which did not recruit other users. Of the actively recruiting class 214 of those users were picked to be seed recruiters while 50 were not. Users were separated into three (3) classes based on their realized recruitment counts: the first class was for users who recruited 3 or fewer recruits (1441 class members), the second class was for users who recruited between 4 and 6 recruits (37 class members), and the third class was for users who recruited 7 or more recruits (135 class members). For testing the model the dataset was randomly shuffled and split into a training set of 800 users, a validation set of 400 users, and a testing set of 443 users.

| Recruit Counts | Participants |
|---|---|
| 0 | 1379 |
| 1 | 24 |
| 2 | 9 |
| 3 | 28 |
| 4 | 19 |
| 5 | 17 |
| 6 | 31 |
| 7 | 129 |
| 8 | 5 |
| 14 | 1 |

Fig. 1. Recruitment counts by the number of participants who achieved these counts

For each user, demographic information and entry survey data was collected; a total of 69 features were constructed from this information for use with our experiments. This data ranged from race, gender, age, educational status, marital status, social network, to a wide range of smoking habit-related questions.

### B. Neural Network Model Setting

We used a run-of-the-mill neural network with the following architecture:

- Input Layer of 69 nodes.
- First Hidden Layer of 100 nodes.
- Second Hidden Layer of 100 nodes.
- Logistic Regression Layer of one node per class.
- Softmax + Argmax Layer for single class output.

The model was trained using negative log likelihood as a cost function and the training process was terminated after the cost over the validation subset ceased to improve significantly. The model was then tested over the test set for our results in the experiments section.

The number of nodes at each hidden layer were selected empirically. In the model construction phase, more analysis will be done in future for a more optimal neural network architecture. This model is tried-and-true method and provides a baseline for our investigation into our problem.

### C. Results and Discussion

In the first experimental run, we achieved an accuracy over the testing set of 95.93%. However, this hides the strength of the model due to the high quantity testing accuracy of the first class. More importantly, we achieved a testing in-class accuracy for the third class of 88.89%. Given that the third class corresponds to users who maximized the recruitment potential of the entire dataset, having a high accuracy for this class over the others is highly desirable.

In the second experimental run, with the same parameters but a different shuffle of the dataset, the model had an overall testing accuracy of 96.16% and an in-class accuracy of 93.54% for the third class.

To ensure that our choice of class labels was reasonable, additional experiments were performed with a relabeling of

| | Class 1 | Class 2 | Class 3 |
|---|---|---|---|
| Training | 704 | 32 | 64 |
| Validation | 345 | 20 | 35 |
| Testing | 392 | 15 | 36 |

Fig. 2. Original Labeling, First Run, Counts

| | Class 1 | Class 2 | Class 3 | Total |
|---|---|---|---|---|
| Training | 98.58 | 56.25 | 90.63 | 96.25 |
| Validation | 97.97 | 40.00 | 88.57 | 94.25 |
| Testing | 98.21 | 53.33 | **88.89** | **95.93** |

Fig. 3. Original Labeling, First Run, Accuracies

| | Class 1 | Class 2 | Class 3 |
|---|---|---|---|
| Training | 693 | 35 | 72 |
| Validation | 353 | 15 | 32 |
| Testing | 395 | 17 | 31 |

Fig. 4. Original Labeling, Second Run

| | Class 1 | Class 2 | Class 3 | Total |
|---|---|---|---|---|
| Training | 98.56 | 74.29 | 80.56 | 95.88 |
| Validation | 97.45 | 66.67 | *100.00* | 96.50 |
| Testing | 96.96 | 82.35 | **93.54** | **96.16** |

Fig. 5. Original Labeling, Second Run, Accuracies

the dataset. In the first additional experiment (Relabeling 1), the first class included users who recruited 2 or fewer, the second class between 3 and 4, and the third class 5 or more. Testing accuracy for this experiment was 86.23%, with a 38.80% in-class accuracy for the third class. In this and all proceeding relabeling experiments, note that the testing accuracy has different baselines and is heavily biased by the size of the worst-recruiting class. Closer attention should be paid to the in-class accuracies of each class.

| Recruit Counts | 0 - 2 | 3 - 4 | 5+ |
|---|---|---|---|
| Training | 700 | 19 | 81 |
| Validation | 342 | 11 | 47 |
| Testing | 371 | 17 | 55 |

Fig. 6. Relabeling 1

| | 0 - 2 | 3 - 4 | 5+ | Total |
|---|---|---|---|---|
| Training | 98.42 | *0.00* | 41.98 | 90.37 |
| Validation | 97.95 | *0.00* | 42.55 | 88.75 |
| Testing | 99.46 | *0.00* | **45.45** | **88.94** |

Fig. 7. Relabeling 1, Accuracies

In the second additional experiment (Relabeling 2), we separated the first class into additional class for users who recruited 0 recruits, keeping the other classes the same. In this experiment, we achieved 90.74% testing accuracy and 58.54% in-class accuracy for the best-recruitment class. While we did perform better in both the in testing accuracy and in-class accuracy of the best-recruitment class, we did not achieve even comparable accuracy with that of the original class labelings.

| Recruit Counts | 0 | 1 - 2 | 3 - 4 | 5+ |
|---|---|---|---|---|
| Training | 664 | 16 | 26 | 94 |
| Validation | 331 | 12 | 9 | 48 |
| Testing | 385 | 5 | 12 | 41 |

Fig. 8. Relabeling 2

| | 0 | 1 - 2 | 3 - 4 | 5+ | Total |
|---|---|---|---|---|---|
| Train. | 98.49 | *0.00* | *0.00* | 53.19 | 88.00 |
| Valid. | 97.58 | *0.00* | *0.00* | 58.33 | 87.75 |
| Test. | 98.18 | *0.00* | *0.00* | **58.54** | **90.74** |

Fig. 9. Relabeling 2, Accuracies

The previous two relabelings tested the hypothesis that potentially the model could classify more precise binnings of the dataset. We've shown empirically that this is not the case. The next two relabelings will test the hypothesis that the model is able to increase performance by reducing the problem to a binary classification problem of only two classes.

In the third additional experiment (Relabeling 3), the first class consisted of users who recruited 3 or fewer recruits and the second class consisted of users who recruited 4 or more recruits. This resulted in a testing accuracy of 89.61% and an in-class accuracy of 27.78% for the best-recruitment class.

| Recruit Counts | 0 - 3 | 4+ |
|---|---|---|
| Training | 708 | 92 |
| Validation | 344 | 56 |
| Testing | 389 | 54 |

Fig. 10. Relabeling 3

| Recruit Counts | 0 - 3 | 4+ | Total |
|---|---|---|---|
| Training | 97.88 | 46.74 | 92.00 |
| Validation | 98.55 | 46.43 | 91.25 |
| Testing | 98.20 | **27.78** | **89.61** |

Fig. 11. Relabeling 3, Accuracies

In the fourth additional experiment (Relabeling 4), the first class consisted of users who recruited 6 or fewer recruits and the second class consisted of users who recruited 7 or more recruits. This resulted in a testing accuracy of 92.10% and an in-class accuracy of 2.94% for the best-recruitment class.

| Recruit Counts | 0 - 6 | 7+ |
|---|---|---|
| Training | 741 | 59 |
| Validation | 358 | 42 |
| Testing | 409 | 34 |

Fig. 12. Relabeling 4

| Recruit Counts | 0 - 6 | 7+ | Total |
|---|---|---|---|
| Training | 100.00 | 3.39 | 92.88 |
| Validation | 99.72 | 4.76 | 89.75 |
| Testing | 99.51 | **2.94** | **92.10** |

Fig. 13. Relabeling 4, Accuracies

As we can see from Relabelings 3 and 4, binary classification is a more difficult problem if our end-goal is to increase accurate classification of users in the best-recruitment class. Even though the best-recruitment class in Relabeling 4 was the same as in the original experiment, we performed worse by orders of magnitude.

## IV. Conclusion

In this paper we presented our idea to achieve a higher classification accuracy of the highly-sought class by sacrificing the accuracy of other unimportant classes. This high accuracy was only achieved when the model was trained under the guise of a multiclass classification problem, separating the samples of unimportant classes into incidental and irrelevant classes. We saw that this approach outperforms the binary classification scheme for the same highly-sought class. This is generally contrary to popular intuition that binary classification is an easier problem. In our future work, we will explore this phenomena in-depth.

## V. Acknowledgement

Funding for this study was received from the National Cancer Institute grant: R21 (R21CA158968). Dr. Sadasivam is funded by a National Cancer Institute Career Development Award (K07CA172677).

## References

[1] S. Haykin. Neural networks: A comprehensive foundation. *MacMillan Publishing Company*, 1994.
[2] T. Mitchell. Machine learning. *McGraw-Hill*, 1997.
[3] R. S. Sadasivam, Cutrona SL, Volz E, Rao SR, and Houston TK. Web based peer driven chain referrals for smoking cessation. *Stud Health Technol Inform*, 2013;192:357-61.
[4] R. S. Sadasivam, M. Erik Volz, L. Rebecca Kinney, R. Sowmya Rao, and K. Thomas Houston. Share2quit: Web-based peer-driven referrals for smoking cessation. *JMIR Res Protoc*, 2(2):e37, Sep 2013.
[5] R.S. Sadasivam, Cutrona SL, Luger TM, Volz E, Kinney R, Rao SR, Allison JJ, and Houston TK. Share2quit: Online social network peer marketing of tobacco cessation systems. *Nicotine Tob Res.*, 2016 Jul 22.
[6] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016.