On Multi-Class Extensions of Adjusted Classify and Count

Mirko Bunse[0000−0002−5515−6278]

Artificial Intelligence Unit, TU Dortmund University, 44227 Dortmund, Germany
mirko.bunse@cs.tu-dortmund.de

Abstract. Adjusted Classify and Count (ACC) is one of the most widely acknowledged methods for quantification, the supervised learning task of predicting the class prevalences in a data sample. While ACC stems from binary quantification, where only two classes are considered, several different multi-class extensions have been proposed. In this work, we compare four existing multi-class extensions, both conceptually and empirically. Moreover, we propose a novel multi-class extension that employs an un-constrained least squares optimization with the aid of a soft-max layer. Our empirical results on a recent benchmark data set demonstrate that numerical optimization techniques for multi-class ACC, like our proposed method, outperform analytic solutions.

Keywords: Quantification · Multi-class classification · Constrained optimization · Unconstrained optimization

1 Introduction

Quantification [8] is the task of predicting the prevalence of each class in a data sample. This supervised learning task is in contrast to "standard" classification learning, where predictions for individual data items, and not for a sample of items, are desired to be accurate. Applications of quantification arise in text sentiment analyses [9], in the social sciences [11], in astroparticle physics [4], and in several other areas.

One of the most widely acknowledged methods for quantification is the *Adjusted Classify and Count* (ACC) technique [8], which was initially proposed for binary quantification in particular. For the multi-class setting, there are at least four different extensions to binary ACC: one-versus-all decomposition [8], matrix inversion [12,17], pseudo-inversion [14], and constrained least squares [3,7,11]. ACC has desireable properties, such as Fisher consistency [16] and computational efficiency.

In this work, we discuss the four existing alternatives and we propose a novel multi-class ACC extension. Our proposal employs un-constrained least squares with the aid of a soft-max layer. We compare the five multi-class extensions empirically on a gold-standard benchmark from the LeQua2022 competition [6]. Our reusable implementation is available online.[1]

[1] https://github.com/mirkobunse/QUnfold.jl

Sec. 2 introduces binary ACC and Sec. 3 presents the multi-class extensions. Our experiments are discussed in Sec. 4 before we conclude in Sec. 5.

2 Adjusted Classify and Count in Binary Quantification

In the following, we revisit four fundamental methods for *binary* quantification, where predictions $\hat{y}_i \in \{-1, +1\}$ take one of two values. In the binary setting, the goal is to predict the prevalence of the positive class, $\mathbb{P}(Y = +1)$, in a sample with N data items. We start with the un-adjusted methods before we detail the adjustment rule that yields consistent quantifiers.

First, the (un-adjusted) Classify and Count (CC) method [8] estimates the prevalence $\mathbb{P}(Y = +1)$ from predictions that are issued for each individual data item in a sample, i.e.

$$p^{(\mathrm{CC})} \;=\; \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}_{\hat{y}_i = +1} \,. \tag{1}$$

At this point, the crisp predictions $\hat{y}_i$ can be replaced with estimates of the posterior probabilities [2], which several classification methods return as an indicator of uncertainty. This proposal leads to the (un-adjusted) Probabilistic Classify and Count (PCC) estimate

$$p^{(\mathrm{PCC})} \;=\; \frac{1}{N} \sum_{i=1}^{N} \widehat{\mathbb{P}}\big(\widehat{Y} = +1 \mid X = \vec{x}_i\big) \,. \tag{2}$$

CC and PCC are easily extended to multi-class settings, where each component $[\vec{p}]_i$ of a vector $\vec{p} \in \mathbb{R}^C$ estimates the prevalence of one class $i \in \{1, \ldots, C\}$, as according to Eq. 1 or Eq. 2.

Unfortunately, it is well-acknowledged that CC and PCC are susceptible to prior probability shift [8, 16], due to imperfections of the underlying classifier. In particular, Eqs. 1 and 2 will systematically over- or under-estimate the true class prevalences if these prevalences deviate from the ones that are used during the training of the classifier. In quantification, these prevalences are typically not known a priori, so that prior probability shift must be expected. Therefore, CC and PCC are not appropriate solutions for the quantification problem.

In binary quantification, we can correct this deficiency through an adjustment rule. This rule leads to the *Adjusted Classify and Count* (ACC) method [8], one of the most widely acknowledged techniques for handling prior probability shift in quantification. Binary ACC estimates $\mathbb{P}(Y = +1)$ as

$$p^{(\mathrm{ACC})} \;=\; \frac{p^{(\mathrm{CC})} - \mathrm{FPR}}{\mathrm{TPR} - \mathrm{FPR}}, \tag{3}$$

where $\mathrm{FPR} = \mathbb{P}(\widehat{Y} = +1 \mid Y = -1)$ is the false positive rate of the underlying classifier and $\mathrm{TPR} = \mathbb{P}(\widehat{Y} = +1 \mid Y = +1)$ is the true positive rate. Both of these rates need to be estimated on hold-out data that is not used during the

training of the classifier; otherwise, overfitting of $p^{(\mathrm{ACC})}$ is likely. If the number of falsely positive predicted instances and the number of falsely negative predicted instances are equal, one can return $p^{(\mathrm{CC})}$ without making an adjustment.

The adjustment rule from Eq. 3 can also be applied to $p^{(\mathrm{PCC})}$ instead of $p^{(\mathrm{CC})}$. In this case, the adjustment rule yields the *Probabilistic Adjusted Classify and Count* (PACC) method [2].

Binary ACC and PACC have desireable properties. Most importantly, they are Fisher consistent estimators of $\mathbb{P}(Y = +1)$ even under prior probability shift [16]. Moreover, they are computationally efficient: a prediction requires only a single pass over the data sample to compute $p^{(\mathrm{CC})}$ or $p^{(\mathrm{PCC})}$; so does the computation of FPR and TPR during training. Hence, multi-class extensions to the binary ACC and PACC are promising topics for quantification research.

3 Multi-Class Extensions of Adjusted Classify and Count

In multi-class quantification with $C > 2$ classes, the goal is to estimate a vector $\vec{p} \in \mathcal{P}$ of class prevalences, where the set of feasible solutions

$$\mathcal{P} \;=\; \left\{ \vec{p} \in \mathbb{R}^C \;:\; [\vec{p}]_i \geq 0 \;\forall\; 1 \leq i \leq C \;\;\wedge\;\; 1 = \sum_{i=1}^{C} [\vec{p}]_i \right\} \tag{4}$$

is the unit simplex. All solutions within this set are valid probability densities.

In the following, we detail four existing multi-class extensions of ACC. We further propose one additional extension, an un-constrained least squares estimate which employs a soft-max layer.

Tab. 1 displays a summary of the conceptual properties of these extensions. In this table, we emphasize that each row respectively extends its preceeding row only in terms of a single aspect. Therefore, we recognize all of these methods as being "true" ACC extensions, rather than being independent methods.

Table 1. Adjustments in multi-class ACC extensions.

adjustment	basis	loss function	constraints	optimization
one-vs-rest (Eq. 5)	TPR_i, FPR_i	—	—	—
inverse (Eq. 8)	M	—	—	—
pseudo-inverse (Eq. 9)	M	least squares	min. norm	—
constrained (Eq. 10)	M	least squares	$\mathcal{P}$	constrained
soft-max (Eq. 11)	M	least squares	$\mathcal{P}$	unconstrained

3.1 One Versus Rest Decomposition

The most straightforward extension of binary ACC decomposes the multi-class quantification problem into C one-versus-rest tasks [8]. Each of these tasks requires a binary quantification of one class versus all others. Hence, we can use the

binary adjustment rule from Eq. 3 in each of the tasks separately. The resulting estimate is $\vec{p}^{\,(\text{one-vs-rest})} \in \mathbb{R}^C$, where

$$[\vec{p}^{\,(\text{one-vs-rest})}]_i \;=\; \frac{[\vec{p}^{\,(\text{CC})}]_i - \text{FPR}_i}{\text{TPR}_i - \text{FPR}_i} \tag{5}$$

is the i-th component of $\vec{p}^{\,(\text{one-vs-rest})}$. Here, $[\vec{p}^{\,(\text{CC})}]_i$ is the CC estimate for the i-th class. Moreover, TPR_i and FPR_i are the true positive rate and the false positive rate when class i is classified against all other classes.

Like in binary ACC, the estimate from Eq. 5 requires clipping to ensure that each component is between 0 and 1. Moreover, this estimate requires normalization to ensure that the sum of all components is one. Unfortunately, these ad-hoc corrections can lead to estimation errors if the data sets are not sufficiently large to accurately estimate $[\vec{p}^{\,(\text{CC})}]_i$, TPR_i, and FPR_i.

3.2 Matrix Inversion

A multi-class classifier can confuse each pair of classes with a non-zero probability. The confusion matrix $M \in \mathbb{R}^{C\times C}$ of a classifier comprises all of these probabilities in the matrix cells

$$[M]_{ij} \;=\; \mathbb{P}\big(\widehat{Y} = i \mid Y = j\big). \tag{6}$$

The matrix of ground-truth confusion probabilities, which are typically unknown, defines the CC outcome

$$\vec{p}^{\,(\text{CC})} \;=\; M \cdot \vec{p}, \tag{7}$$

from the ground-truth prevalence vector $\vec{p} \in \mathcal{P}$. Consequently, we can recover an estimate of the true $\vec{p}$ with an estimate of the confusion matrix M.

The most straightforward attempt in this direction [12,17] is to invert an estimate of M to yield the prevalence estimate

$$\vec{p}^{\,(\text{inverse})} \;=\; M^{-1} \cdot \vec{p}^{\,(\text{CC})}. \tag{8}$$

For instance, this matrix inversion estimate is implemented in the current release[2] of QuaPy [13]. Since QuaPy is likely the most complete and usable software package for quantification, this choice has established $\vec{p}^{\,(\text{inverse})}$ as the "quasi-standard" multi-class extension of ACC and PACC.

However, the inverse of an estimated M is not guaranteed to exist. In this case, the estimator is undefined. QuaPy deals with this issue by falling back to the un-adjusted $\vec{p}^{\,(\text{CC})}$ and $\vec{p}^{\,(\text{PCC})}$ if M is not invertible.

3.3 Pseudo-Inversion

A robust alternative to matrix inversion replaces the actual inverse M^{-1} with the Moore-Penrose pseudo-inverse $M^{\dagger}$. This replacement leads to the estimate

$$\vec{p}^{\,(\text{pseudo-inverse})} \;=\; M^{\dagger} \cdot \vec{p}^{\,(\text{CC})}, \tag{9}$$

[2] QuaPy, v0.1.6: https://github.com/HLT-ISTI/QuaPy/releases/tag/0.1.6

which is always defined because $M^\dagger$ is always guaranteed to exist. Moreover, $M^\dagger$ is equal to M^{-1} if M^{-1} exists. Hence, the replacement does not reduce the quality of the estimate. It gains robustness because no fallback to an un-adjusted $\vec{p}^{\,(\mathrm{CC})}$ or $\vec{p}^{\,(\mathrm{PCC})}$ is necessary if M is not invertible.

The pseudo-inverse estimator is proven to be a least-squares estimate of the true $\vec{p}$, which is constrained to the minimum norm estimate [14, Th. 4.1]. This constraint has the advantage that $\vec{p}^{\,(\mathrm{pseudo-inverse})}$ is unique. However, a minimum norm constraint lacks motivation from a practical perspective; in fact, the constraint is unrelated to the actual feasible set $\mathcal{P}$ from Eq. 4.

3.4 Constrained Least Squares

Both inversion techniques $\vec{p}^{\,(\mathrm{inverse})}$ and $\vec{p}^{\,(\mathrm{pseudo-inverse})}$ suffer from not being constrained to the feasible set $\mathcal{P}$ from Eq. 4. In fact, both techniques tend to produce estimates that i) do not sum to one and ii) have components that are less than zero. This deficiency is typically addressed through clipping and normalization, an ad-hoc correction that can lead to estimation errors.

A more appropriate approach is presented by Hopkins and King [11]. They propose a constrained optimization task

$$\vec{p}^{\,(\mathrm{constrained})} \;=\; \underset{\vec{p}\in\mathcal{P}}{\arg\min} \left\| \vec{p}^{\,(\mathrm{CC})} - M \cdot \vec{p} \right\|_2^2, \tag{10}$$

which explicitly constrains the estimate to the space $\mathcal{P}$ of valid probabilities. Within this space, the most accurate estimate according to the L_2 norm is searched for. Hence, $\vec{p}^{\,(\mathrm{constrained})}$ employs the same loss function as the estimate $\vec{p}^{\,(\mathrm{pseudo-inverse})}$, but uses a more appropriate set of constraints.

Unfortunately, Hopkins and King [11] do not propose a specific algorithm to solve Eq. 10. While an analytical solution exists for the unit sum constraint $1 = \sum_{i=1}^{C} [\vec{p}]_i$ [1, Chap. 1.4], we are not aware of an analytic solution that considers the inequality constraints $[\vec{p}]_i \geq 0 \;\forall\; 1 \leq i \leq C$ from Eq. 4.

Consequently, the optimization of Eq. 10 requires numerical optimization techniques. In our implementation, we employ a primal-dual interior-point algorithm with a filter line search [18]. However, other numerical methods are conceivable at this point. For instance, Firat [7] employs a sequential quadratic programming technique [19, Chap. 18] to solve Eq. 10. We leave a comparison of numerical optimization techniques in quantification to future work.

3.5 Unconstrained Least Squares with a Soft-Max Layer

We now propose a novel multi-class extension of binary ACC. The goal of our proposal is to rephrase the optimization task from Eq. 10 to achieve an unconstrained optimization task which produces valid probability densities despite being unconstrained. The desire to optimize without constraints is rooted in our subjective perception that unconstrained optimization is an easier problem than constrained optimization. Hence, we hope for less noise in the gradients that are computed during the optimization process.

We obtain an unconstrained optimization task through a soft-max layer, which is a derivable operation that transforms latent variables into probability densities. We maintain the least squares loss function from Eq. 10. Our multi-class ACC is defined over latent variables $\vec{l} \in \mathbb{R}^C$, as

$$\begin{aligned} \vec{p}^{\,(\mathrm{soft-max})} &= \mathrm{softmax}(\vec{l}^{\,*}), \\ \vec{l}^{\,*} &= \operatorname*{arg\,min}_{\vec{l} \in \mathbb{R}^C} \left\| \vec{p}^{\,(\mathrm{CC})} - M \cdot \mathrm{softmax}(\vec{l}) \right\|_2^2 + \lambda \cdot \left\| \vec{l} \right\|_2^2, \\ [\mathrm{softmax}(\vec{l})]_i &= \frac{\exp([\vec{l}]_i)}{\sum_{j=1}^{C} \exp([\vec{l}]_j)}, \end{aligned} \tag{11}$$

where $\lambda \cdot \|\vec{l}\|_2^2$ is a regularization term that ensures all $\exp([\vec{l}]_i)$ to be finite within floating point precision. This regularization term is only a technical detail: it affects the latent variables $\vec{l}$, but not the estimate $\vec{p}^{\,(\mathrm{soft-max})}$, which is always in $\mathcal{P}$ due to the soft-max layer. In our experiments, we fix $\lambda = 10^{-6}$.

4 Experiments

In the following, we intend to uncover the merits and the disadvantages of the above multi-class extensions of ACC and PACC. To this end, we evaluate their performance on the public data set [5] of the LeQua2022 competition [6]. Our reusable Julia implementation of methods and experiments is available online.[1]

The LeQua2022 dataset is designed to constitute a gold-standard benchmark, both for binary text quantification and for multi-class text quantification. The multi-class problem in this competition features 28 classes, 20 000 training items and 1 000 validation samples. Each of the validation samples consists of 1 000 data items that are drawn according to varying class prevalences. We employ the vectorial representation of the data and a logistic regression classifier, which obtained the highest performance on this representation during the competition [15]. We optimize the regularization parameter of this classifier on the validation set and over the grid $\{10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}\}$, to obtain the best performance for each quantification method. The selection of the best regularization parameter is either in terms of the absolute error (AE) or in terms of the relative absolute error (RAE). We report the results, in terms of both metrics, on the test set.

All multi-class extensions of ACC require the estimation of the confusion matrix M (or at least the rates TPR_i and FPR_i) on hold-out data. In order to use all labeled data for classifier training and for the adjustments, we use a bagging ensemble of size 100. We estimate M, TPR_i, and FPR_i on the out-of-bag predictions of this ensemble.

During the hyper-parameter optimization on the validation set, almost all methods succeeded in producing estimates for the class prevalences. An exception to this outcome is the matrix inversion from Eq. 8 in ACC. This method failed to produce prevalence estimates for the values 10^{-3} and 10^{-2} of the classifier's

regularization parameter because these values led to confusion matrices M that were *not* invertible.

Table 2. Test set performance of the different multi-class adjustments, for ACC and PACC and in terms of AE and RAE. The performance of the best adjustment in each setting is printed in boldface.

adjustment	ACC		PACC	
	AE	RAE	AE	RAE
un-adjusted (Eq. 1 / Eq. 2)	0.0254	2.5532	0.0246	2.6771
one-vs-rest (Eq. 5)	0.0262	4.1484	0.0262	4.1484
inverse (Eq. 8)	0.0222	1.7224	0.0195	1.5288
pseudo-inverse (Eq. 9)	0.0177	1.7224	0.0195	1.5288
constrained (Eq. 10)	0.0158	1.2826	0.0123	**0.9908**
soft-max (Eq. 11)	**0.0130**	**1.2633**	**0.0106**	1.0886

Discussion The results from Tab. 2 demonstrate that the different multi-class adjustments exhibit quite different performances, in general. The lowest errors are achieved by the constrained estimator from Eq. 10 (in terms of RAE in PACC) and by our unconstrained soft-max estimator from Eq. 11 (in terms of all other configurations). The margins of improvement over all other adjustments are considerable: for instance, the constrained PACC achieves an RAE that is 38% smaller than the RAE of the pseudo-inverse PACC (last column, 0.9669 vs 1.5536); our soft-max PACC achieves an AE that is 46% smaller than the AE of the pseudo-inverse PACC (third column, 0.0106 vs 0.0197).

5 Conclusions and Outlook

We have discussed five different multi-class extensions of the binary adjustment that is employed in ACC and PACC. One of these extensions is an original proposal by us; this proposal employs a soft-max layer to circumvent the constraints that are otherwise required to obtain valid solutions in a numerical optimization process. Our proposal and an existing constrained least squares adjustment [3,7,11] deliver the most competitive performances.

Future work should compare different optimization techniques [19] to solve the constrained optimization task and our unconstrained soft-max proposal. Our "trick" of using a soft-max layer in quantification is also applicable to other methods, like ReadMe [11] and HDx / HDy [10], where it should be evaluated.

References

1. Amemiya, T.: Advanced econometrics. Blackwell (1985)

2. Bella, A., Ferri, C., Hernández-Orallo, J., Ramírez-Quintana, M.J.: Quantification via probability estimators. In: Int. Conf. on Data Mining. pp. 737–742. IEEE (2010). https://doi.org/10.1109/ICDM.2010.75
3. Bunse, M., Morik, K.: Unification of algorithms for quantification and unfolding. In: Workshop on Mach. Learn. for Astropart. Phys. and Astron. Gesellschaft für Informatik e.V. (2022), to appear
4. Bunse, M., Piatkowski, N., Morik, K., Ruhe, T., Rhode, W.: Unification of deconvolution algorithms for Cherenkov astronomy. In: Int. Conf. on Data Sci. and Adv. Anal. pp. 21–30. IEEE (2018). https://doi.org/10.1109/DSAA.2018.00012
5. Esuli, A., Moreo, A., Sebastiani, F.: Learning to quantify: LeQua 2022 datasets (2021). https://doi.org/10.5281/zenodo.6546188
6. Esuli, A., Moreo, A., Sebastiani, F.: A detailed overview of LeQua@CLEF 2022: Learning to quantify. In: Conf. and Labs of the Eval. Forum. pp. 1849–1868. CEUR Workshop Proc. (2022), http://ceur-ws.org/Vol-3180/paper-146.pdf
7. Firat, A.: Unified framework for quantification. arXiv:abs/1606.00868 (2016)
8. Forman, G.: Quantifying counts and costs via classification. Data Mining and Knowl. Discov. **17**(2), 164–206 (2008). https://doi.org/10.1007/s10618-008-0097-y
9. Gao, W., Sebastiani, F.: From classification to quantification in tweet sentiment analysis. Soc. Netw. Anal. and Mining **6**(19), 1–22 (2016)
10. González-Castro, V., Alaíz-Rodríguez, R., Alegre, E.: Class distribution estimation based on the Hellinger distance. Inf. Sci. **218**, 146–164 (2013). https://doi.org/10.1016/j.ins.2012.05.028
11. Hopkins, D.J., King, G.: A method of automated nonparametric content analysis for social science. Amer. J. of Polit. Sci. **54**(1), 229–247 (2010). https://doi.org/10.1111/j.1540-5907.2009.00428.x
12. McLachlan, G.J.: Discriminant analysis and statistical pattern recognition. Wiley (1992)
13. Moreo, A., Esuli, A., Sebastiani, F.: Quapy: A python-based framework for quantification. In: Int. Conf. on Inf. and Knowl. Management. pp. 4534–4543. ACM (2021). https://doi.org/10.1145/3459637.3482015
14. Mueller, J.L., Siltanen, S.: Linear and Nonlinear Inverse Problems with Practical Applications, Computational science and engineering, vol. 10. SIAM (2012). https://doi.org/10.1137/1.9781611972344
15. Senz, M., Bunse, M.: DortmundAI at LeQua 2022: Regularized SLD. In: Conf. and Labs of the Eval. Forum. CEUR Workshop Proc., vol. 3180, pp. 1911–1915 (2022), http://ceur-ws.org/Vol-3180/paper-152.pdf
16. Tasche, D.: Fisher consistency for prior probability shift (2017)
17. Vucetic, S., Obradovic, Z.: Classification on data with biased class distribution. In: Eur. Conf. on Mach. Learn. pp. 527–538. Springer (2001). https://doi.org/10.1007/3-540-44795-4_45
18. Wright, S.J., Nocedal, J.: Numerical optimization. Operations Research and Financial Engineering, Springer, 2 edn. (2006)
19. Wächter, A., Biegler, L.T.: On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Math. programming **106**(1), 25–57 (2006)