Improving Model Probability Calibration by Integration of Large Data Sources with Biased Labels

Renat Sergazinov ^{1}, Richard Chen ^{2}, Cheng Ji ^{2}, Jing Wu ^{3}, Daniel Cociorva ^{2}, Hakan Brunzell ^{2}
^{1}Department of Statistics, Texas A&M University, College Station, TX USA
^{2}Buyer Risk Prevention, Amazon, San Diego, CA USA
^{3} University of Illinois Urbana-Champaign, Champaign, IL USA
mrsergazinov@tamu.edu, {rychardy, cjiamzn, cociorva, brunzell}@amazon.com

Abstract

Probability calibration transforms raw output of a classification model into empirically interpretable probability. When the model is purposed to detect rare event and only a small expensive data source has clean labels, it becomes extraordinarily challenging to obtain accurate probability calibration. Utilizing an additional large cheap data source is very helpful, however, such data sources oftentimes suffer from biased labels. To this end, we introduce an approximate expectation-maximization (EM) algorithm to extract useful information from the large data sources. For a family of calibration methods based on the logistic likelihood, we derive closed-form updates and call the resulting iterative algorithm CalEM. We show that CalEM inherits convergence guarantees from the approximate EM algorithm. We test the proposed model in simulation and on the real marketing datasets, where it shows significant performance increases.

1 Introduction

Machine learning models, including neural networks and gradient boosted trees, often suffer from calibration issues (Zadrozny and Elkan 2001; Guo et al. 2017; Zadrozny and Elkan 2001; Kuleshov, Fenner, and Ermon 2018; Fernández et al. 2018). Calibration refers to the alignment of a model's predicted probabilities with the actual likelihood of outcomes. For instance, if a model predicts a probability of 0.2 for a specific class over 100 instances, ideally, 20 of those instances should belong to that class. This aspect of model performance is crucial for model trustworthiness and safety. There has been significant research on developing metrics for calibration assessment (Nixon et al. 2019; Gupta et al. 2020; Gruber and Buettner 2022), methods to post-correct mis-calibrated models (Platt et al. 1999; Zadrozny and Elkan 2002; Kumar, Liang, and Ma 2019; Gupta et al. 2020), and techniques for creating better-calibrated classifiers (Bohdal, Yang, and Hospedales 2023).

In this paper, we tackle a separate challenge of training a probability calibration model under practical constraints of label imbalance and small sample size. Under such constraints, accurate probability calibration becomes challenging as we illustrate in Figure 1. As the sample size decreases and the label imbalance exacerbates, the variance (calibration error) of the probability calibration model increases significantly.

A prominent example of real-world data with severe class imbalance and small sample size comes from the marketing literature (Diemert et al. 2018; Ke et al. 2021; Liu et al. 2023). In this context, a key objective is to estimate the probability that a customer will naturally convert to a given brand. With accurate estimates, marketing firms can strategically allocate budgets to target the most promising customer segments and improve overall sales. The target variable, a binary conversion indicator, is highly imbalanced (often with fewer than 1% converting), and the only unbiased dataset is the control group that receives no marketing intervention – data which can be both costly and limited. Meanwhile, the treatment group, exposed to marketing campaigns, produces biased labels unsuitable for direct modeling (Ke et al. 2021). Similar scenarios arise in (1) medical modeling, where the target is disease susceptibility and the treatment is a vaccine; (2) fraud prevention, where the target is fraudulent activity and the treatment is a preventative measure; and (3) online education, where the target is course completion and the treatment is a learning intervention. In all of these cases, the goal is to estimate the target, but the control data may be expensive, while treated samples have modified outcomes and cannot be used directly.

To solve this challenge, we propose a modified EM algorithm to fit any likelihood-based calibration model on the combined control and treatment portions of the dataset. We first estimate the calibration on the control data and then proceed to estimating the transition probability between the treatment and control sets. The calibration model could then be re-fit on the combined dataset with weights computed according to the transition probabilities. The algorithm then proceeds by iteratively refining the transition probability and calibration model estimates. As we illustrate in Figure 1, the algorithm results in smaller variance (calibration error). We theoretically show that our approach could be formulated as an approximate EM (Dempster, Laird, and Rubin 1977) and demonstrate convergence guarantees. We empirically validate its performance in simulation and on real marketing datasets, where we show improvement over the baseline methods fitted on the control-only data.

The rest of the paper is structured as follows. In Section 3 we introduce a general EM algorithm with approximate E-

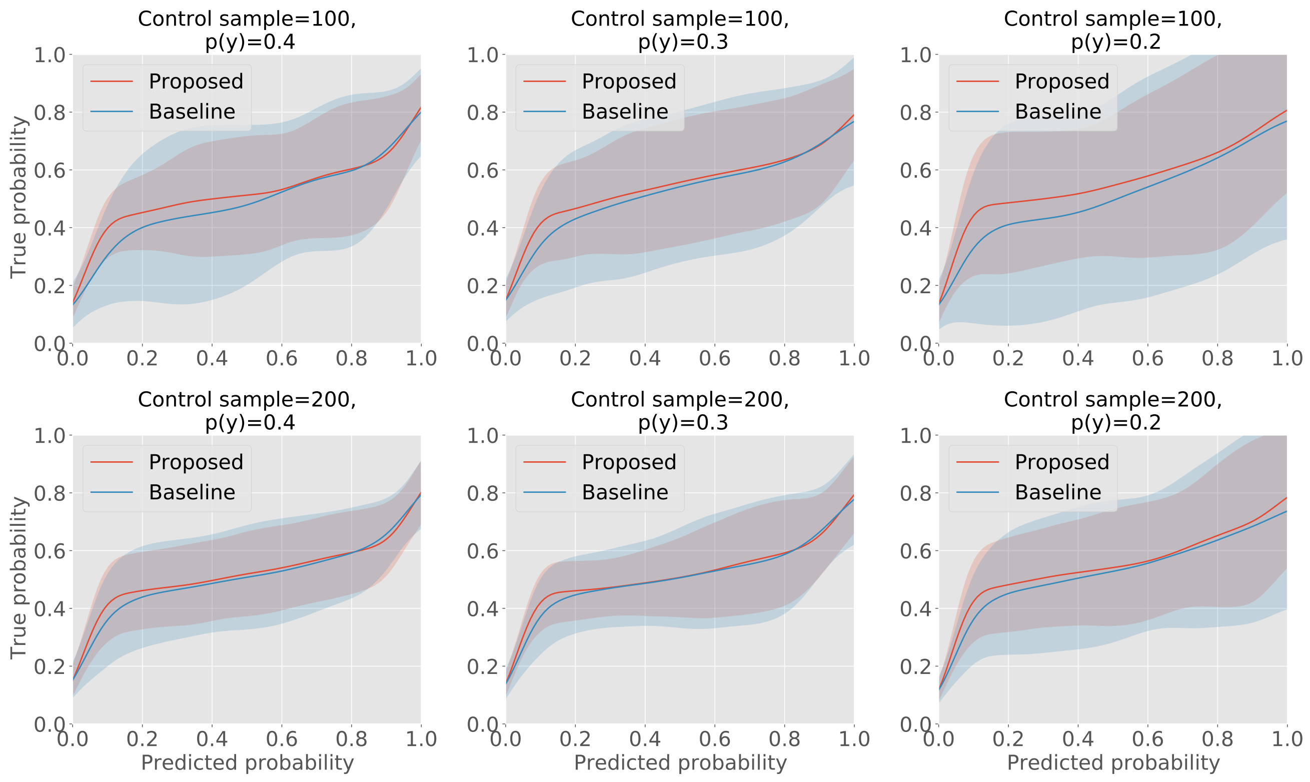


Figure 1: Variance of the probability calibration models in simulation. Note that larger variance leads to higher expected calibration error, by bias-variance decomposition. The shaded area represents the 95% confidence intervals around the curves. The baseline model is fitted on the small control data, while our proposed approach uses EM to fit on the combined control and large noisy datasets. Total sample size of the combined dataset is 2,000.

step which has an application to data augmentation for model calibration, we also show the convergence property of this approximate EM algorithm. We introduce baseline calibration in Section 4.2, and design data-augmented calibration algorithm based on the approximate EM algorithm in Section 4.3. In Section 5 we demonstrate our data-augmented model calibration in an empirical study. We put the proof and derivation in Appendix.

2 Related Work

Probability Calibration A variety of calibration techniques has been proposed in the literature such as works by Groeneboom and Lopuhaa (1993); Platt et al. (1999); Zadrozny and Elkan (2001); Niculescu-Mizil and Caruana (2005b,a); Naeini, Cooper, and Hauskrecht (2015); Kull, Silva Filho, and Flach (2017); Kull et al. (2019); Kumar, Liang, and Ma (2019); Gupta et al. (2021); van der Laan et al. (2023). Key desiderata for calibration methods is to be accuracy-preserving and data-efficient. To achieve the first, calibration methods are usually constrained to be monotonic. The latter has been studied by Kumar, Liang, and Ma (2019), who established sample efficiency for various calibration methods.

In practice, probability calibration can be integrated during model training (Platt et al. 1999; Niculescu-Mizil and Caruana 2005a) or applied as a post-hoc process (Groeneboom and Lopuhaa 1993; Kull, Silva Filho, and Flach 2017; Gupta et al. 2021). The post-hoc approach is generally favored in practical applications due to its superior performance and modularity, which allows the base classification model to remain unaltered.

Despite their popularity, post-hoc calibration models suffer from the need to maintain a separate calibration set for fitting. As we show in Figure 1, the variance (calibration error) of the model is highly dependent on the class imbalance and sample size. In many practical scenarios, maintaining a large calibration data set could be prohibitively expensive. In this work, we propose an algorithm that could be utilized together with any existing likelihood-based calibration models to incorporate large biased data source that substantially reduces the calibration error, whilst also not requiring the access to the expensive unbiased data.

Latent Variables To treat the bias in the large data source, we use the latent variable formulation. Numerous statistical frameworks are designed to handle unobserved (latent) variables effectively. For cases involving missing data, inverse probability weighting methods have demonstrated excellent performance, supported by robust theoretical guar-

Algorithm 1: EM with approximate E-step

```
**Initialization**: specify initial point $\theta^{(0)}$, tolerance $\varepsilon$; and learn $\eta$;
**Expectation-maximization iterations** for $t = 0, 1, 2, \cdots$:
  1. implement E-step by
       $Q_\eta(\theta|\theta^{(t)}) = \mathbb{E}_{\mathcal{U}\sim P_{\theta^{(t)},\eta}(\mathcal{U}|\mathcal{O})}\big[\log P_\theta(\mathcal{O},\mathcal{U})\big];$
  2. implement M-step by
       $\theta^{(t+1)} = \operatorname*{argmax}_\theta Q_\eta(\theta|\theta^{(t)});$
  3. continue until $\big\|\theta^{(t+1)} - \theta^{(t)}\big\| < \varepsilon$.
```

$$\begin{tikzcd}
P_{\theta^{(t)}} \arrow[r, squiggly, "\mathrm{EM}"] \arrow[d, leftrightarrow, "d_0"'] & P_{\theta^*} \arrow[dl, leftrightarrow, "d_1 > d_0"] \\
P_{\theta^{(t)},\eta} &
\end{tikzcd}$$

Figure 2: Intuition of the convergence of our approximate EM

antees (Wooldridge 2007). In our scenario, we have a small control set that is representative of the population, alongside a larger dataset where a biased proxy of the response variable is available. The work in Chatterjee et al. (2016) explores a similar issue, utilizing a large additional data source to estimate the distribution of covariates in a regression framework. The study in Yang and Ding (2020) aligns even more closely with our setup, with the key difference being the adoption of a causal inference framework: the authors are interested in estimating the causal effects. We draw inspiration from these methodologies to develop our algorithm, which we build on the EM framework (Dempster, Laird, and Rubin 1977).

3 Approximate EM Algorithm

Let θ be the parameter of interest, $\mathcal{O}$ be the observed data, and $\mathcal{U}$ be the unobserved (latent) data. We can analytically formulate the log-likelihood function $\log P_\theta(\mathcal{O},\mathcal{U})$ with respect to both observed and unobserved data. However, without the unobserved data $\mathcal{U}$, one cannot analytically formulate the log-likelihood function $\mathcal{L}_\mathcal{O}(\theta) = \log P_\theta(\mathcal{O})$ with respect to observed data $\mathcal{O}$ alone. EM algorithm circumvents this difficulty, it computes expectation of the log-likelihood function $\log P_\theta(\mathcal{O},\mathcal{U})$ conditional on observed data, and then maximizes the conditional expectation.

Given a starting value of parameter, the EM algorithm alternates the following two steps iteratively:

- **E-step:** find the conditional distribution of latent variables given observed variables and current parameter value, and compute the conditional expectation of the full likelihood of both observed and latent data;

Algorithm 2: EM with approximate E-step for penalized likelihood

```
**Initialization**: specify penalty hyperparameter $\lambda$, initial point $\theta^{(0)}$, tolerance $\varepsilon$; and learn $\eta$;
**Expectation-maximization iterations** for $t = 0, 1, 2, \cdots$:
  1. implement E-step by
       $Q_{\lambda,\eta}(\theta|\theta^{(t)}) = \mathbb{E}_{\mathcal{U}\sim P_{\theta^{(t)},\eta}(\mathcal{U}|\mathcal{O})}\big[\log P_\theta(\mathcal{O},\mathcal{U})\big] - \Omega_\lambda(\theta);$
  2. implement M-step by maximizing the penalized expected log-likelihood function:
       $\theta^{(t+1)} = \operatorname*{argmax}_\theta Q_{\lambda,\eta}(\theta|\theta^{(t)});$
  3. continue until $\big\|\theta^{(t+1)} - \theta^{(t)}\big\| < \varepsilon$.
```

- **M-step:** maximize the conditional expectation of the full likelihood and update the parameter value.

The E-step and M-step of the EM algorithm can be described by

$$Q(\theta|\theta^{(t)}) \;:=\; \mathbb{E}_{\mathcal{U}\sim P_{\theta^{(t)}}(\mathcal{U}|\mathcal{O})}\big[\log P_\theta(\mathcal{O},\mathcal{U})\big], \tag{1}$$

$$\theta^{(t+1)} \;=\; \operatorname*{argmax}_\theta Q(\theta|\theta^{(t)}), \tag{2}$$

where $P_{\theta^{(t)}}(\mathcal{U}|\mathcal{O})$ is the conditional distribution of latent variables conditioning on observed variables, parameterized by $\theta^{(t)}$. The EM algorithm can reach a local maximum of $\mathcal{L}_\mathcal{O}(\theta)$ (or the global maximum if $\mathcal{L}_\mathcal{O}(\theta)$ is concave), even though the log-likelihood $\mathcal{L}_\mathcal{O}(\theta)$ is not computable.

In practice, the E-step of Equation (1) may be difficult to compute. The probabilistic model of latent variable conditional on observed data may be so complicated that it is prohibitive to express the expectation in Equation (1) analytically. We propose an approximate E-step in which we use a surrogate conditional distribution $P_{\theta^{(t)},\eta}(\mathcal{U}|\mathcal{O})$ as an approximation of the exact conditional distribution $P_{\theta^{(t)}}(\mathcal{U}|\mathcal{O})$ in iteration t. The additional parameter η in the surrogate is learned from the large noisy dataset and does not need to be updated in EM iterations.

If we model $P_{\theta,\eta}$ wisely, the E-step in Algorithm 1 is conveniently implementable. In addition, if the noisy data size is sufficient, $P_{\theta^{(t)},\eta}(\mathcal{U}|\mathcal{O})$ can be a close proxy to ensure convergence. Below we give a theoretical result with regard to the convergence property of our approximate EM algorithm. The proof of the theorem can be found in Appendix A.1.

Theorem 1. *Suppose θ^* is a local maximum of the likelihood function. Assume we can learn the nuisance parameter η so that $P_{\theta,\eta}$ is sufficiently close to P_θ in the following sense:*

$$D_{\mathrm{KL}}\big(P_{\theta,\eta}(\cdot|\mathcal{O})\,\big|\,P_{\theta^*}(\cdot|\mathcal{O})\big) \geq D_{\mathrm{KL}}\big(P_{\theta,\eta}(\cdot|\mathcal{O})\,\big|\,P_\theta(\cdot|\mathcal{O})\big), \tag{3}$$

as long as $\|\theta - \theta^*\| < \gamma$, where $D_{\mathrm{KL}}(\cdot|\cdot)$ is Kullback–Leibler divergence[1] *and γ is a positive constant, then Algorithm 1 (EM algorithm with approximate E-step) converges.*

[1]Kullback–Leibler divergence is defined as $D_{\mathrm{KL}}(p|q) =$

Intuitively speaking, as long as the surrogate conditional distribution is closer to the exact conditional distribution than to the true conditional distribution, our approximate EM algorithm converges. The Requirement (3) is not a stringent assumption and we provide its intuition in Figure 2. Note that the Requirement (3) is trivially true when $\theta = \theta^*$.

Modern practices of maximum likelihood estimation often incorporate penalty, i.e., choose

$$L_\lambda(\theta) = \mathcal{L}_{\mathcal{O}}(\theta) - \Omega_\lambda(\theta) \tag{4}$$

as the objective function, where Ω_λ is a penalty function with hyperparameter λ. The purpose of penalizing log-likelihood is to achieve certain desired properties, e.g. sparsity, monotonicity, etc. For probability calibration in Section 4, we need a penalty term to impose a shape constraint on calibration function. We apply the idea of EM algorithm with approximate E-step to penalized maximum likelihood estimation in Algorithm 2.

We can extend the convergence property of EM algorithm with approximate E-step in Theorem 1 to penalized maximum likelihood estimation.

Corollary 1. *Suppose θ^* is a local maximum of (4). Assume we can learn the nuisance parameter η such that $P_{\theta,\eta}$ satisfies (3) as long as $\|\theta - \theta^*\| < \gamma$ for a positive constant γ, then Algorithm 2 converges.*

4 Data-Augmented Model Calibration

4.1 Notation and Setup

Let $\mathbb{X}$ be a feature space and $\mathbb{Y} = \{0, 1\}$ be a binary label – we label a sample that belongs to the target category as 1 (positive), otherwise we label it as 0 (negative). Denote the random variables corresponding to features and labels by $\boldsymbol{X}$ and Y respectively. We call the variable Y unbiased or control. In contrast, denote by Z the biased or treatment variables, which have distribution $p_Z \neq p_Y$. Denote a classification model by $f : \mathbb{X} \mapsto [0, 1]$, i.e. $f(\boldsymbol{X})$ is raw model score. The raw score $f(\boldsymbol{X})$ from most modern machine learning models does not represent empirical probability. To calibrate raw model output whilst retain its predictive capability, a monotonically increasing function g_θ is used to post hoc process raw model output as $g_\theta \circ f(\boldsymbol{X})$, where θ is a multi-dimensional parameter. We call this function g_θ a probability calibration model.

4.2 Vanilla Probability Calibration

As we outline in Section 2, a probability calibration model g_θ can be learned in a variety of ways. In this work in order to utilize EM algorithm, we focus on the models g_θ that can be learned via logistic regression. Specifically, we focus on the case when:

$$Y|f(\boldsymbol{X}) \sim \text{Bernoulli}\left(g_\theta(f(\boldsymbol{X}))\right) \tag{5}$$

$$g_\theta(w) = \frac{1}{1 + \exp\{-s_\theta(w)\}}. \tag{6}$$

$E_p[\log(p/q)]$. It can be interpreted as a distance metric in space of probability distributions – it measures distance between probability distributions.

For example, Kull, Silva Filho, and Flach (2017) propose to parameterize s_θ as a bi-variate function to make the logistic objective correspond to optimizing Beta distribution. Gupta et al. (2021) parameterize s_θ as a monotonic spline function.

In particular, this probabilistic formulation allows us to directly write the estimation procedure as a likelihood optimization, which in turn allows us to introduce the EM step. Since we focus on the calibration methods that can be learned via logistic regression objective, in the rest of this paper, we generically denote a learning algorithm for g_θ by

$$g_\theta \leftarrow \text{LogReg}(\mathcal{D}),$$

where $\mathcal{D}$ are the observations of $(\boldsymbol{X}, Y)$.

Probability Calibration: Case Study In marketing data, the target category is customer conversion. A machine learning model is used to score the probability of converting a customer. The overall goal is to use the model to target customers with higher likelihood of converting, which would simultaneously save the resources whilst retaining the overall effectiveness. The majority of the traffic (usually $> 90\%$) is subject to the marketing treatment. The remaining small portion of traffic is exempt from any treatment to serve as holdout or control group for performance measurement and calibration purposes. We denote small clean data from the minority group by $\mathcal{D}_S$ and large noisy data from the majority group by $\mathcal{D}_B$:

$$\begin{aligned} &\text{small clean dataset:} && \mathcal{D}_S = \{\boldsymbol{x}_l, y_l\}_{l \in \mathcal{S}}, \\ &\text{large noisy dataset:} && \mathcal{D}_B = \{\boldsymbol{x}_k, z_k\}_{k \in \mathcal{B}}, \end{aligned}$$

where $\boldsymbol{x}_l$ and $\boldsymbol{x}_k$ are feature vectors, y_l is an unbiased label, and z_k is the observed but potentially biased label.

Note that the treated sample labels $\{z_k\}_{k \in \mathcal{B}}$ may contain bias. Intuitively, we want to decouple the default probability of customer conversion from the effect of the marketing campaign. In the treatment group, these two effects are confounded, so vanilla probability calibration can only utilize the small clean dataset $\mathcal{D}_S$. In the cases of severe label imbalance, this could lead to especially large variance and instability. Therefore, it is of interest to incorporate information from the biased sample $\mathcal{D}_B$ into fitting the calibration model.

4.3 Data-Augmented Calibration via Approximate EM Algorithm

We augment the data by incorporating latent variable in the following way

$$\text{large noisy dataset:} \quad \{\boldsymbol{x}_k, z_k, y_k\}_{k \in \mathcal{B}},$$

where $\{y_k\}_{k \in \mathcal{B}}$ are latent counterfactual labels (would-be labels if no treatment was applied).

Under this formulation, the observed data and unobserved data become

$$\mathcal{O} = \mathcal{D}_S \cup \mathcal{D}_B, \qquad \mathcal{U} = \{y_k\}_{k \in \mathcal{B}}. \tag{7}$$

When the cardinality of clean labels is far less than that of corrupted labels, i.e. $|\mathcal{S}| \ll |\mathcal{B}|$, data augmentation combining $\mathcal{D}_S$ and $\mathcal{D}_B$ and extracting information from large biased labels provides significant improvement over vanilla probability calibration on $\mathcal{D}_S$.

The hurdle to utilize the large noisy data is treatment-induced bias in observed labels $\{z_k\}_{k\in\mathcal{B}}$. We consider the unobserved clean labels $\{y_k\}_{k\in\mathcal{B}}$ of samples in the large noisy data as latent information. We use the approximate EM algorithm introduced in Section 3 to learn probability calibration in the presence of latent data.

We formulate $P_\theta(\mathcal{U}|\mathcal{O})$ for our use cases as follows. Let Z denote the label from a customer that has undergone treatment. There are two possible outcomes:

- If a sample has a negative label, we suppose that the treatment did not sufficiently alter the outcome. Hence, we assume that the counterfactual outcome, Y, would still be negative if there was no treatment;
- If a sample has undergone a treatment and the outcome is positive, with a non-zero probability the positive outcome could have been caused by the treatment. In other words, counterfactual outcome, Y, could have been positive if there was no treatment.

We formalize our intuition in the following statement.

Assumption 1 (Treatment effect)**.**

$$P(Y=0|Z,\boldsymbol{X}) = \begin{cases} 1 & \textit{if } Z=0 \\ h(\boldsymbol{X}) & \textit{if } Z=1 \end{cases}. \quad (8)$$

We can find such function $h:\mathbb{X}\mapsto[0,1]$ *as long as* $P(Y=0|f(\boldsymbol{X})=0)<1$.[2]

Now let us deduce the form of the function h. Using the law of total probability and Bayes's theorem, we write

$$P(Y=0|\boldsymbol{X}) = P(Z=0|\boldsymbol{X}) + h(\boldsymbol{X})\cdot P(Z=1|\boldsymbol{X}).$$

Therefore, we have that:

$$h(\boldsymbol{X}) = \frac{P(Y=0|\boldsymbol{X}) - P(Z=0|\boldsymbol{X})}{P(Z=1|\boldsymbol{X})}. \quad (9)$$

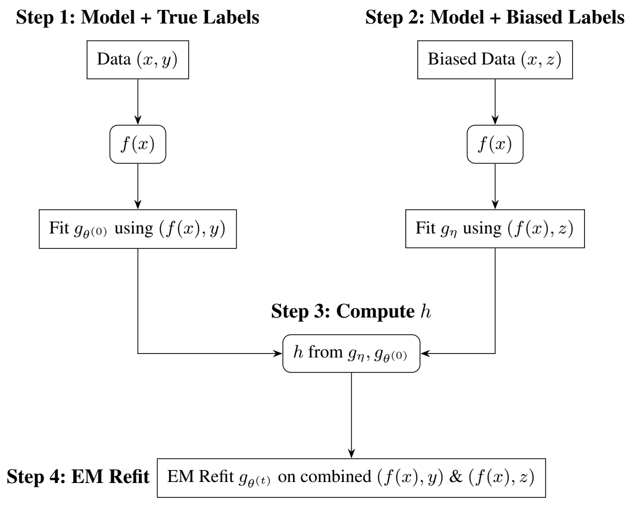
Note that $h(\boldsymbol{X})$ is a well-defined probability measure: $h(\boldsymbol{X})\in[0,1]$ because $P(Z=0|\boldsymbol{X})\le P(Y=0|T=1,\boldsymbol{X})<1$, $\forall\boldsymbol{X}\in\mathbb{X}$.

Based on the Assumption (1) and Algorithm 1, we propose a new algorithm called CalEM. We describe CalEM in Algorithm 3. The full derivation of Algorithm 3 can be found in Appendix A.2. We also provide a visual illustration in Figure 3. The algorithm works by first fitting a calibrator $g_{\theta^{(0)}}$ on the model's predictions and true labels, and a second calibrator g_η on the same predictions and biased labels. Next, it computes a transition function h from $g_{\theta^{(0)}}$ and g_η, and then uses an expectation-maximization (EM) approach to iteratively refit $g_{\theta^{(t)}}$ on the combined dataset, guided by h. This process adjusts for label bias and improves the calibration of the model's predictions.

5 Experiments

First, we investigate the performance of the proposed approach under controlled settings in simulations in Section 5.1. We then apply our approach to the real marketing datasets in Section 5.2.

[2]The condition $P(Y=0|f(\boldsymbol{X})=0)<1$ is not stringent. On the contrary, $P(Y=0|f(\boldsymbol{X})=0)=1$ means the classifier $f(\boldsymbol{X})=0$ predicts negative label with 100% accuracy.



Step 1: Model + True Labels
Step 2: Model + Biased Labels
Data (x, y)
Biased Data (x, z)
$f(x)$
$f(x)$
Fit $g_{\theta^{(0)}}$ using $(f(x), y)$
Fit g_η using $(f(x), z)$
Step 3: Compute h
h from $g_\eta, g_{\theta^{(0)}}$
Step 4: EM Refit
EM Refit $g_{\theta^{(t)}}$ on combined $(f(x), y)$ & $(f(x), z)$

Figure 3: Summary of the CalEM algorithm

5.1 Simulation

In simulation our primary goals twofold: test how the model performance changes with respect to the observed data proportion, event probability, and the choice of transition and calibration functions, h and g respectively. As a baseline model, we choose spline-based calibration model (Gupta et al. 2021), which we denote by GAM. First, we first fit GAM on the clean sample. Then, we enhance GAM with CalEM and re-fit on the large biased dataset.

For simulation, we sample true probabilities from $Beta(\alpha,\beta)$ distribution, where α,β allows us to control the event probability. To imitate the effect of the miscalibrated classifier, we transform the true probabilities to model scores via function, g^{-1}. The treatment effect is simulated via the transition function, h. Overall, our sampling models for generating the data is:

$$\begin{aligned} p_i &\sim Beta(\alpha,\beta), \\ z_i &\sim Bernoulli(p_i), \\ \tilde{p}_i &= p_i - h(p_i) * p_i, \\ y_i|p_i,z_i &\sim \begin{cases} 0 & \text{if } z_i=0 \\ Bernoulli(1-h(p_i)) & \text{if } z_i=1 \end{cases}. \end{aligned}$$

In total, we generate $10,000$ samples and select some portion of the data to be the clean dataset, $\mathcal{D}_S=\{g^{-1}(\tilde{p}_i),y_i\}$, and the rest to be the noisy data, $\mathcal{D}_B=\{g^{-1}(\tilde{p}_i),z_i\}$.

For the miscalibration curve g^{-1}, we explore two design options: $g_1^{-1}(p)=\frac{1}{1+\exp\{-20(p-0.5)\}}$, which represents an overconfident classifier that pushes scores to the extremes, and $g_2^{-1}(p)=p^3$, which corresponds to a classifier trained on imbalanced data, tending to push scores closer to 0. Similarly, for the transition function, we consider two options: $h_1(p)=0.5p^2$ and $h_2(p)=0.3\times\mathbf{1}\{p\ge0.3\}$, where the primary goal is to assess how the smoothness of the transition function impacts the estimation procedure.

For each simulation setup, we train the baseline GAM model on the observed portion, $\mathcal{D}_S$. We then fit the proposed model on the combined dataset, $\mathcal{D}_S\cup\mathcal{D}_B$. As a metric, we

	$\% / P(Y)$	g_1				g_2			
		0.5	0.4	0.3	0.2	0.5	0.4	0.3	0.2
h_1	10%	$+35.55\%^*$	$+24.49\%^*$	$+32.00\%^*$	$+11.26\%^*$	$+21.62\%^*$	$+47.51\%^*$	$+46.96\%^*$	$+52.82\%^*$
	20%	$+21.14\%^*$	$+18.68\%^*$	$+19.45\%^*$	$+7.17\%$	$+23.97\%^*$	$+33.13\%^*$	$+48.55\%^*$	$+32.87\%^*$
	30%	$+20.12\%^*$	$+24.23\%^*$	$+23.18\%^*$	-3.62%	$+18.79\%^*$	$+12.91\%$	$+52.48\%^*$	$+33.40\%^*$
	40%	$+12.49\%^*$	$+12.31\%^*$	$+10.30\%^*$	$+15.46\%^*$	$+15.03\%^*$	$+22.21\%$	$+27.56\%^*$	$+38.04\%^*$
	50%	$+4.94\%$	$+7.12\%^*$	$+2.12\%$	$+4.82\%$	$+8.58\%^*$	$+4.46\%$	$+20.08\%^*$	$+25.09\%^*$
h_2	10%	$+3.95\%^*$	$+2.62\%^*$	$+4.77\%^*$	$+2.94\%$	$+4.93\%^*$	$+12.23\%^*$	$+31.84\%^*$	$+17.34\%^*$
	20%	$+7.98\%^*$	$+6.56\%^*$	$+10.32\%^*$	$+8.92\%^*$	$+19.60\%^*$	$+23.98\%^*$	$+46.98\%^*$	$+43.49\%^*$
	30%	$+7.32\%^*$	$+4.27\%^*$	$+5.03\%^*$	$+4.55\%^*$	$+11.63\%^*$	$+14.61\%^*$	$+28.72\%^*$	$+29.33\%^*$
	40%	$+3.39\%^*$	$+8.04\%^*$	$+1.44\%$	$+0.79\%$	$+6.56\%^*$	$+20.18\%^*$	$+15.64\%^*$	$+28.13\%^*$
	50%	$+3.95\%^*$	$+2.62\%^*$	$+4.77\%^*$	$+2.94\%$	$+4.93\%^*$	$+12.23\%^*$	$+31.84\%^*$	$+17.34\%^*$

Table 1: Simulation results: average percent improvement in L^2-error of the proposed model over the baseline GAM fit on the observed-only data. We indicate with star (*) the results that are statistically significant at the 5% level using paired T-test.

	Criteo			Hillstorm			Lenta		
	KS	Brier	Log-lik.	KS	Brier	Log-lik.	KS-error	Brier	Log-lik.
GAM	0.000489	0.001656	0.008835	0.016997	0.092418	0.326527	0.002450	0.077768	0.270673
IR	$\underline{0.000199}$	0.001653	0.008229	0.016958	0.092875	0.343015	0.002484	0.077877	0.271399
BC	0.000224	$\underline{0.001647}$	$\underline{0.008089}$	$\underline{0.015970}$	$\underline{0.091948}$	$\underline{0.324656}$	$\underline{0.002073}$	0.077755	0.270462
GAM-EM	0.000389	0.001650	0.008773	0.016290	0.092224	0.325976	0.002357	**0.077708**	$\underline{0.270407}$
BC-EM	**0.000190**	**0.001643**	**0.008083**	**0.015884**	**0.091926**	**0.324489**	**0.001918**	$\underline{0.077715}$	**0.270340**

Table 2: Results on the marketing datasets with 10 re-runs. The best results are highlighted in **bold**; second best results are underlined. We measure Kolmogorov-Smirnov error (Gupta et al. 2020), Brier score (Brier 1950), and log-likelihood. Our EM approach consistently improves on the baseline method.

compute L^2-error between the oracle calibration curve, g, and the estimated calibration curves. To demonstrate the utility of our approach, we compute the percent improvement as $(L^2_{GAM} - L^2_{EM})/(L^2_{GAM}) * 100\%$. We report the results in Table 1. We observe the proposed method outperform the baseline under all settings. We re-run the simulation 100 times and perform a paired T-test on the L^2 errors. We find that at 5% significance level, our proposed approach achieves a lower L^2 error under most scenarios.

Based on Table 1, we hypothesize that datasets that would benefit the most from the proposed approach are the ones with mild label imbalance, have small observed data size compared to the latent data, and where the influence function h is suspected to be relatively smooth. For the imbalanced datasets, the variance of the calibration curve is heterogenuous and high; therefore, the calibration error improves from adding more data. However, if the observed data sample size is large enough, then there is no room for improvement, so the proposed approach may not yield a significant boost. Regarding smoothness, since we use GAM as our baseline which is piece-wise polynomial, this dictates a smoothness assumption on h by (9). Finally, the shape of the true calibration curve does not seem to affect our method.

5.2 Real Data Application

For our real data analysis, we compare our proposed approach against the mainstream calibration methods: Beta calibration (Kull, Silva Filho, and Flach 2017), isotonic regression (Groeneboom and Lopuhaa 1993), and GAM (Gupta et al. 2021).

We use the datasets Criteo (Diemert et al. 2018), Hillstrom (Hillstrom 2008), and Lenta (Lenta 2020). For all datasets, the goal is to develop a well-calibrated classifier to predict the treatment outcome. Each dataset contains a feature set, a target column, and a binary treatment indicator. Criteo (Diemert et al. 2018) consists of data of 13 million users, each one represented by 12 features with the overall treatment ratio of 84.6%. Hillstrom (Hillstrom 2008) consists of the records of 64,000 users, described by 8 features with the overall treatment ratio of 0.6. Finally, Lenta (Lenta 2020) contains information on 687,000 users, described by 193 features with the overall treatment ratio of 75%.

To test the proposed approach, we randomly split the observed portion of the dataset, consisting of untreated users, into training, calibration, and test sets with a ratio of $90\% - 5\% - 5\%$, respectively. We use the training dataset to train the XGBoost classifier (Chen and Guestrin 2016). We use the calibration set for fitting the calibration model. Finally, we test the calibration on the test set, measuring Kolmogorov-Smirnov error (Gupta et al. 2020), Brier score

```
Algorithm 3: CalEM
Input: read a small clean dataset $\mathcal{D}_S$ + a large noisy dataset $\mathcal{D}_B$, and specify tolerance $\varepsilon$;
Process: Construct dataset
    $\mathcal{D}^* = \mathcal{D}_S \cup \mathcal{D}_B \cup \{\boldsymbol{x}_k, y_k = 0\}_{k \in \mathcal{B}}$;
Initialization: learn an initial calibration function $g_{\theta^{(0)}}$ using $\mathcal{D}_S$ and a calibration function on the biased dataset using $\mathcal{D}_B$ using logistic regression objective, i.e.,
    $g_{\theta^{(0)}} \leftarrow \mathrm{LogReg}(\mathcal{D}_S) \quad g_\eta \leftarrow \mathrm{LogReg}(\mathcal{D}_B)$;
Computing an initial estimate of "treatment effect":
    $h_{\theta^{(0)},\eta}(\boldsymbol{x}_j) = \frac{g_\eta(f(\boldsymbol{x}_j)) - g_{\theta^{(0)}}(f(\boldsymbol{x}_j))}{g_\eta(f(\boldsymbol{x}_j))}$;
Expectation-maximization iterations for $t = 0, 1, 2, \cdots$:
  1. compute weight vector:
       $\boldsymbol{w}^{(t)} = \left(\mathbf{1}_{|\mathcal{S}|}^{\mathrm{T}}, \boldsymbol{w}_1^{(t),\mathrm{T}}, \boldsymbol{w}_2^{(t),\mathrm{T}}\right)^{\mathrm{T}}$,
     where T denotes vector transpose, and
     • $\mathbf{1}_{|\mathcal{S}|}$ is a column vector of 1's whose dimension equals the cardinality of $\mathcal{S}$, i.e. $|\mathcal{S}|$,
     • $\boldsymbol{w}_1^{(t)}$ is a column vector of dimension $|\mathcal{B}|$, and its $j$-th element is $\boldsymbol{w}_{1,j}^{(t)} = 1 - h_{\theta^{(t)},\eta}(\boldsymbol{x}_j)$,
     • $\boldsymbol{w}_2^{(t)}$ is a column vector of dimension $|\mathcal{B}|$, and its $j$-th element is $\boldsymbol{w}_{2,j}^{(t)} = h_{\theta^{(t)},\eta}(\boldsymbol{x}_j)$;
  2. update calibration function $g_{\theta^{(t)}} \to g_{\theta^{(t+1)}}$ by:
       $g_{\theta^{(t+1)}} \leftarrow \mathrm{LogReg}(\mathcal{D}^*; \boldsymbol{w}^{(t)})$;
  3. update "treatment effect" estimate:
       $h_{\theta^{(t+1)},\eta}(\boldsymbol{x}_j) = \frac{g_\eta(f(\boldsymbol{x}_j)) - g_{\theta^{(t+1)}}(f(\boldsymbol{x}_j))}{g_\eta(f(\boldsymbol{x}_j))}$;
  4. continue until $\left\|\theta^{(t+1)} - \theta^{(t)}\right\| < \varepsilon$.
```

(Brier 1950), and log-likelihood. We report the mean error results across 10 runs in Table 2. As we outline in Section 4.2 both Beta calibration (Kull, Silva Filho, and Flach 2017) and GAM (Gupta et al. 2020) are likelihood-based approaches. Following Algorithm 3, we combine them with our proposed CalEM method. Upon analyzing the results, we find that CalEM consistently enhances the performance of the baseline algorithms and achieves optimal performance across all metrics. These outcomes underscore the effectiveness of CalEM in incorporating additional biased data in fitting the calibration model.

6 Conclusion

In this paper, we introduce an iterative approach based on a modified EM algorithm that allows the incorporation of large, biased data sources into the estimation of the probability calibration function. The proposed algorithm can serve as a drop-in enhancement for any likelihood-based recalibration method. Additionally, the method benefits from convergence guarantees inherited from the EM formulation. The significant performance improvements demonstrated in both simulated environments and real-world marketing datasets underscore the effectiveness and practical applicability of the proposed model in enhancing the accuracy of probability calibration under a variety of conditions.

Acknowledgments

Parts of this work were completed during Renat Sergazinov's internship at Amazon. Renat Sergazinov would like to express sincere gratitude to the Amazon Buyer Risk Prevention team for their support, guidance, and invaluable feedback throughout the project. Their collective expertise and encouragement were instrumental in shaping the direction and depth of this research.

References

Bohdal, O.; Yang, Y.; and Hospedales, T. 2023. Meta-Calibration: Learning of Model Calibration Using Differentiable Expected Calibration Error. *TMLR*.

Brier, G. W. 1950. Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78(1): 1–3.

Chatterjee, N.; Chen, Y.-H.; Maas, P.; and Carroll, R. J. 2016. Constrained maximum likelihood estimation for model calibration using summary-level information from external big data sources. *Journal of the American Statistical Association*, 111(513): 107–117.

Chen, T.; and Guestrin, C. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 785–794.

Dempster, A. P.; Laird, N. M.; and Rubin, D. B. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 39(1): 1–22.

Diemert, E.; Betlei, A.; Renaudin, C.; and Amini, M.-R. 2018. A large scale benchmark for uplift modeling. In *KDD*.

Fernández, A.; García, S.; Galar, M.; Prati, R. C.; Krawczyk, B.; and Herrera, F. 2018. *Learning from imbalanced data sets*, volume 10. Springer.

Groeneboom, P.; and Lopuhaa, H. 1993. Isotonic estimators of monotone densities and distribution functions: basic facts. *Statistica Neerlandica*, 47(3): 175–183.

Gruber, S.; and Buettner, F. 2022. Better uncertainty calibration via proper scores for classification and beyond. *Advances in Neural Information Processing Systems*, 35: 8618–8632.

Guo, C.; Pleiss, G.; Sun, Y.; and Weinberger, K. Q. 2017. On calibration of modern neural networks. In *International conference on machine learning*, 1321–1330. PMLR.

Gupta, K.; Rahimi, A.; Ajanthan, T.; Mensink, T.; Sminchisescu, C.; and Hartley, R. 2020. Calibration of neural networks using splines. *ICLR*.

Gupta, K.; Rahimi, A.; Ajanthan, T.; Mensink, T.; Sminchisescu, C.; and Hartley, R. 2021. Calibration of neural networks using splines. *ICLR*.

Hillstrom, K. 2008. MineThatData E-Mail Analytics And Data Mining Dataset. Accessed: 2024-08-13.

Ke, W.; Liu, C.; Shi, X.; Dai, Y.; Philip, S. Y.; and Zhu, X. 2021. Addressing exposure bias in uplift modeling for large-scale online advertising. In *2021 IEEE International Conference on Data Mining (ICDM)*, 1156–1161. IEEE.

Kuleshov, V.; Fenner, N.; and Ermon, S. 2018. Accurate uncertainties for deep learning using calibrated regression. In *International conference on machine learning*, 2796–2804. PMLR.

Kull, M.; Perello Nieto, M.; Kängsepp, M.; Silva Filho, T.; Song, H.; and Flach, P. 2019. Beyond temperature scaling: Obtaining well-calibrated multi-class probabilities with dirichlet calibration. *Advances in neural information processing systems*, 32.

Kull, M.; Silva Filho, T.; and Flach, P. 2017. Beta calibration: a well-founded and easily implemented improvement on logistic calibration for binary classifiers. In *Artificial intelligence and statistics*, 623–631. PMLR.

Kumar, A.; Liang, P. S.; and Ma, T. 2019. Verified uncertainty calibration. *Advances in Neural Information Processing Systems*, 32.

Lenta. 2020. BigTarget Hackathon Dataset. Dataset provided for the BigTarget Hackathon, Summer 2020. Accessed: 2024-08-13.

Liu, D.; Tang, X.; Gao, H.; Lyu, F.; and He, X. 2023. Explicit feature interaction-aware uplift network for online marketing. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 4507–4515.

Naeini, M. P.; Cooper, G.; and Hauskrecht, M. 2015. Obtaining well calibrated probabilities using bayesian binning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 29.

Niculescu-Mizil, A.; and Caruana, R. 2005a. Obtaining calibrated probabilities from boosting. *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*, 28–33.

Niculescu-Mizil, A.; and Caruana, R. 2005b. Predicting good probabilities with supervised learning. *Proceedings of the 22nd International Conference on Machine Learning*.

Nixon, J.; Dusenberry, M. W.; Zhang, L.; Jerfel, G.; and Tran, D. 2019. Measuring Calibration in Deep Learning. In *CVPR workshops*, volume 2.

Platt, J.; et al. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3): 61–74.

van der Laan, L.; Ulloa-Pérez, E.; Carone, M.; and Luedtke, A. 2023. Causal isotonic calibration for heterogeneous treatment effects. In *International Conference on Machine Learning*, 34831–34854. PMLR.

Wooldridge, J. M. 2007. Inverse probability weighted estimation for general missing data problems. *Journal of econometrics*, 141(2): 1281–1301.

Yang, S.; and Ding, P. 2020. Combining multiple observational data sources to estimate causal effects. *Journal of the American Statistical Association*, 115(513): 1540–1554.

Zadrozny, B.; and Elkan, C. 2001. Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In *Icml*, volume 1, 609–616.

Zadrozny, B.; and Elkan, C. 2002. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 694–699.

A Appendix

A.1 Convergence of the approximate EM algorithm

In this subsection, we show the EM algorithm with approximate E-step increases the log-likelihood $\mathcal{L}_{\mathcal{O}}(\theta)$ of observed data $\mathcal{O}$, hence the EM algorithm can reach local maximum. If the marginal log-likelihood is concave with respect to the parameter, EM algorithm can reach the global maximum. The following argument is inspired by Chapter 8.4 of (Little and Rubin 2020).

By Bayes's rule,

$$P_\theta(\mathcal{O}) = \frac{P_\theta(\mathcal{O},\mathcal{U})}{P_\theta(\mathcal{U}|\mathcal{O})},$$

so we can write the marginal log-likelihood as

$$\mathcal{L}_{\mathcal{O}}(\theta) = \log P_\theta(\mathcal{O},\mathcal{U}) - \log P_\theta(\mathcal{U}|\mathcal{O}). \tag{10}$$

Take expectation of the both hand sides of (10) with respect to surrogate distribution $P_{\theta^{(t)},\eta}$, we get

$$\mathcal{L}_{\mathcal{O}}(\theta) = \underbrace{\int P_{\theta^{(t)},\eta}(\mathcal{U}|\mathcal{O}) \cdot \log P_\theta(\mathcal{O},\mathcal{U})\, \mathrm{d}\mathcal{U}}_{Q_\eta(\theta|\theta^{(t)})} + \underbrace{\int P_{\theta^{(t)},\eta}(\mathcal{U}|\mathcal{O}) \cdot \log \frac{1}{P_\theta(\mathcal{U}|\mathcal{O})}\, \mathrm{d}\mathcal{U}}_{H\left(P_{\theta^{(t)},\eta}(\cdot|\mathcal{O})\,\middle|\,P_\theta(\cdot|\mathcal{O})\right)}. \tag{11}$$

Note that $Q_\eta(\theta|\theta^{(t)})$ is the objective in iteration t, $H(\cdot|\cdot)$ is cross entropy. Then it follows

$$\begin{aligned}\mathcal{L}_{\mathcal{O}}(\theta) - \mathcal{L}_{\mathcal{O}}(\theta^{(t)}) &= Q_\eta(\theta|\theta^{(t)}) - Q_\eta(\theta^{(t)}|\theta^{(t)}) \\ &\quad - \int P_{\theta^{(t)},\eta}(\mathcal{U}|\mathcal{O}) \cdot \log \frac{P_\theta(\mathcal{U}|\mathcal{O})}{P_{\theta^{(t)},\eta}(\mathcal{U}|\mathcal{O})}\, \mathrm{d}\mathcal{U} \\ &\quad - \int P_{\theta^{(t)},\eta}(\mathcal{U}|\mathcal{O}) \cdot \log \frac{P_{\theta^{(t)},\eta}(\mathcal{U}|\mathcal{O})}{P_{\theta^{(t)}}(\mathcal{U}|\mathcal{O})}\, \mathrm{d}\mathcal{U},\end{aligned}$$

equivalently,

$$\begin{aligned}\mathcal{L}_{\mathcal{O}}(\theta) - \mathcal{L}_{\mathcal{O}}(\theta^{(t)}) = {}& Q_\eta(\theta|\theta^{(t)}) - Q_\eta(\theta^{(t)}|\theta^{(t)}) \\ &+ D_{\mathrm{KL}}\big(P_{\theta^{(t)},\eta}(\cdot|\mathcal{O})\,\big|\,P_\theta(\cdot|\mathcal{O})\big) - D_{\mathrm{KL}}\big(P_{\theta^{(t)},\eta}(\cdot|\mathcal{O})\,\big|\,P_{\theta^{(t)}}(\cdot|\mathcal{O})\big).\end{aligned}$$

According to the assumption (3) in Theorem 1, we have

$$\mathcal{L}_{\mathcal{O}}(\theta) - \mathcal{L}_{\mathcal{O}}(\theta^{(t)}) \geq Q_\eta(\theta|\theta^{(t)}) - Q_\eta(\theta^{(t)}|\theta^{(t)}). \tag{12}$$

(12) shows $\mathcal{L}_{\mathcal{O}}(\theta)$ is guaranteed to improve as long as we optimize $Q_\eta(\theta|\theta^{(t)})$ in each M-step. Hence the EM algorithm with approximate E-step converges.

A.2 Deriving approximate EM algorithm for model calibration

Let $n = |\mathcal{S}|$ and $N = |\mathcal{B}|$, and denote small clean dataset by $(\boldsymbol{x}_j, y_j)$, $j = 1, \cdots, n$ and large noisy dataset by $(\boldsymbol{x}_j, z_j)$, $j = n+1, \cdots, n+N$. $\mathcal{O}$ is the union of these two datasets. Suppose that we have the oracle knowledge of latent data $\mathcal{U} = \{y_j,\ j = n+1, \cdots, n+N\}$, the full log-likelihood function of the calibration parameter θ in (5) incorporating both observed and latent information is

$$\mathcal{L}_{\mathcal{O}\oplus\mathcal{U}}(\theta) = \sum_{j=1}^{n+N} \log L(\theta|y_j, \boldsymbol{x}_j). \tag{13}$$

In view of (8) and $L(\theta|y_j, \boldsymbol{x}_j) = P_\theta(y_j|\boldsymbol{x}_j)$, the objective function (13) can be written as:

$$\begin{aligned}\mathcal{L}_{\mathcal{O}\oplus\mathcal{U}}(\theta) &= \sum_{j=1}^{n} \log L(\theta|y_j, \boldsymbol{x}_j) + \sum_{j=n+1}^{n+N} \log L(\theta|y_j, \boldsymbol{x}_j) \\ &= \sum_{j=1}^{n} \log P_\theta(y_j|\boldsymbol{x}_j) + \sum_{j=n+1}^{n+N} \log P_\theta(y_j|\boldsymbol{x}_j).\end{aligned} \tag{14}$$

However, since we do not observe $\{y_j\}_{j=n+1}^{n+N}$ in the full objective (14), we can not optimize nor compute $\mathcal{L}_{\mathcal{O}\oplus\mathcal{U}}(\theta)$. Here we use the EM algorithm idea and optimize the expected likelihood as:

$$Q(\theta|\theta^{(t)}) = \mathbb{E}_{\mathcal{U}\sim P_{\theta^{(t)}}(\mathcal{U}|\mathcal{O})}\left[\mathcal{L}_{\mathcal{O}\oplus\mathcal{U}}(\theta)\right]$$
$$= \sum_{j=1}^{n} \log P_\theta(y_j|\boldsymbol{x}_j) + \sum_{j=n+1}^{n+N} \mathbb{E}_{y_j\sim P_{\theta^{(t)},\eta}(\cdot|z_j,\boldsymbol{x}_j)}\left[\log P_\theta(y_j|\boldsymbol{x}_j)\right].$$

According to (9), we learn $P_{\theta^{(t)},\eta}(\cdot|z_j,\boldsymbol{x}_j)$ by estimating

$$h_{\theta^{(t)},\eta}(\boldsymbol{x}_j) = \frac{g_\eta(f(\boldsymbol{x}_j)) - g_{\theta^{(t)}}(f(\boldsymbol{x}_j))}{g_\eta(f(\boldsymbol{x}_j))}, \tag{15}$$

where g_η is a separate calibration function fitted on the large dataset, $\mathcal{D}_\mathcal{B}$. The expectation can then be opened as:

$$\begin{aligned}
&\mathbb{E}_{y_j\sim P_{\theta^{(t)},\eta}(\cdot|z_j,\boldsymbol{x}_j)}\left[\log P_\theta(y_j|\boldsymbol{x}_j)\right] = \mathbb{E}_{y_j\sim P_{\theta^{(t)},\eta}(\cdot|z_j,\boldsymbol{x}_j)}\left[y_j \log(g_\theta(s_j)) + (1-y_j)\log(1-g_\theta(s_j))\right]\\
&= (1-z_j)\times[\log(1-g_\theta(s_j))] + z_j \times \left[(1-h_{\theta^{(t)},\eta}(\boldsymbol{x}_j))\log g_\theta(s_j) + h_{\theta^{(t)},\eta}(\boldsymbol{x}_j)\log(1-g_\theta(s_j))\right]\\
&= (1-z_j)\times\left[(1-h_{\theta^{(t)},\eta}(\boldsymbol{x}_j) + h_{\theta^{(t)},\eta}(\boldsymbol{x}_j))\log(1-g_\theta(s_j))\right] +\\
&\qquad z_j\times\left[(1-h_{\theta^{(t)},\eta}(\boldsymbol{x}_j))\log g_\theta(s_j) + h_{\theta^{(t)},\eta}(\boldsymbol{x}_j)\log(1-g_\theta(s_j))\right]\\
&= (1-h_{\theta^{(t)},\eta}(\boldsymbol{x}_j))[(1-z_j)\log(1-g_\theta(s_j)) + z_j\log g_\theta(s_j)] + h_{\theta^{(t)},\eta}(\boldsymbol{x}_j)\log(1-g_\theta(s_j))
\end{aligned} \tag{16}$$

Plugging (16) into $Q(\theta|\theta^{(t)})$, we get:

$$\begin{aligned}
Q(\theta|\theta^{(t)}) &= \sum_{j=1}^{n} 1\times\left[y_j\log g_\theta(s_j) + (1-y_j)\log(1-g_\theta(s_j))\right]\\
&+ \sum_{j=n+1}^{n+N} \underbrace{(1-h_{\theta^{(t)},\eta}(\boldsymbol{x}_j))}_{:=w_{1,j}^{(t)}} \times [z_j\log g_\theta(s_j) + (1-z_j)\log(1-g_\theta(s_j))]\\
&+ \sum_{j=n+1}^{n+N} \underbrace{h_{\theta^{(t)},\eta}(\boldsymbol{x}_j)}_{:=w_{2,j}^{(t)}} \times [\log(1-g_\theta(s_j))],
\end{aligned} \tag{17}$$

which is a weighted with weights $\boldsymbol{w}^{(t)} = \left(\mathbf{1}_{|\mathcal{S}|}^{\mathrm{T}}, \boldsymbol{w}_1^{(t),\mathrm{T}}, \boldsymbol{w}_2^{(t),\mathrm{T}}\right)^{\mathrm{T}}$ version of the original objective.

To start the algorithm, we choose $\theta^{(0)}$ using only clean labels:

$$\theta^{(0)} = \underset{\theta}{\operatorname{argmax}} \sum_{j=1}^{n} \log P_\theta(y_j|\boldsymbol{x}_j).$$

Putting everything together we, therefore, arrive at our algorithm **??**.

Interpretation of the algorithm In view of (17), we have

$$\begin{aligned}
Q(\theta|\theta^{(t)}) &= \underbrace{\text{objective function of GAM}(\mathcal{D}_S;\ \mathbf{1}_{|\mathcal{S}|})}_{y_j\log g_\theta(s_j)+(1-y_j)\log(1-g_\theta(s_j))}\\
&+ \underbrace{\text{weighted objective function of GAM}(\mathcal{D}_B;\ \boldsymbol{w}_1)}_{[1-h_{\theta^{(t)}}(\boldsymbol{x}_j)]\cdot[z_j\log g_\theta(s_j)+(1-z_j)\log(1-g_\theta(s_j))]}\\
&+ \underbrace{\text{weighted objective function of GAM}(\{\boldsymbol{x}_k, y_k=0\}_{k\in\mathcal{B}};\ \boldsymbol{w}_2)}_{h_{\theta^{(t)}}(\boldsymbol{x}_j)\cdot[0\times\log g_\theta(s_j)+1\times\log(1-g_\theta(s_j))]}.
\end{aligned}$$

In other words, the approximate EM algorithm extracts information from the large noisy dataset $\mathcal{D}_B$ iteratively by updating sample weights. Let's pay more attention to the meaning of these sample weights. Note that by definition (8), $h_\theta(X) = P_\theta(Y = 0|Z = 1, X)$.

We can interpret $1 - h_{\theta^{(t)}}(\boldsymbol{x}_j)$ and $h_{\theta^{(t)}}(\boldsymbol{x}_j)$ as our knowledge at iteration t about the relationship between the latent label y_j and observed label z_j. These two sample weights can be interpreted as our confidence in terms of probability.

Intuitively speaking, the objective $Q(\theta|\theta^{(t)})$ combines

1. the original information from small clean data – it is our baseline information;
2. the unmodified information from large noisy data with confidence $P(y_j = z_j)$, we update this confidence in every iteration;
3. “negation” of the information from large noisy data with confidence $P(y_j \neq z_j)$.