Benchmark Algorithms of Distribution Matching

Jim Lim, Ziyu Gong, Brian Ko, Thursday, March 23, 2024

1 Related Papers

List related papers and gives description on how the model is related to distribution matching.

1.1 Calibration

1.1.1 Motivation

Even though many deep learning models are producing high predictive performance, but it is often producing unreliable predictions due to absence of calibration Niculescu-Mizil and Caruana [2005]. Most deep learning models tend to be primarily over confident, this is shown by spiking posterior distribution Wang et al. [2021]. There are several reasons behind this situations such as over-parameterized networks, a lack of regularization, limited data, and imbalanced label distributions.

1.2 Notations

The input $x \in X$ and label $y \in Y = \{1, \ldots, K\}$ are random variables that defines dataset along with dataset size n, $\mathcal{D}_n = \{(x_i, y_i)\}_{i \in [n]}$. Let $\mathcal{H}$ be a model with $h(\mathcal{D}_n) = (\hat{Y}, \hat{P})$, where $\hat{Y}$ is a class prediction and $\hat{P}$ is confidence. We denote calibrator as Q, which maps $Q(\hat{p}) = \hat{q}_i$, where $\hat{q}_i$ is calibrated probability.

1.2.1 Definition of Calibration

Purpose of calibration is to prevent the model's posterior distribution from being either over confident or under confident. **Therefore, we can view this problem as distribution matching by adjusting over or under confident distribution to true distribution.** In classification tasks, a model h is perfectly calibrated on if and only if:

$$P(\hat{Y} = Y | \hat{P} = p) = p, \forall p \in [0, 1] \tag{1}$$

where p is true correctness likelihood. Intuitively, we want to have confidence $\hat{P}$ to be equals to true probability p.

1.2.2 Calibration Metrics

Even though there are various metrics out there to compute the calibration metrics, we can narrow it down to two fundamental metrics since other metrics are just variation from these two metrics.

Reliability Diagram The figure 1 shows reliability diagrams, which are visual representation of calibration. The X axis is corresponding to confidence $\hat{P}$, and Y axis is corresponding to accuracy p. If the model is perfectly calibrated, we have identify function h.

We can calculate accuracy and confidence within bins by grouping prediction into M interval bins. We group together model predictions (confidence) within interval $I_m = (\frac{m-1}{M}, \frac{m}{M}]$. EX) M=4 then $I_m = (0, 0.25], (0.25, 0.5], (0.5, 0.75], (0.75, 1]$, and call it as B_m. We can calculate accuracy by

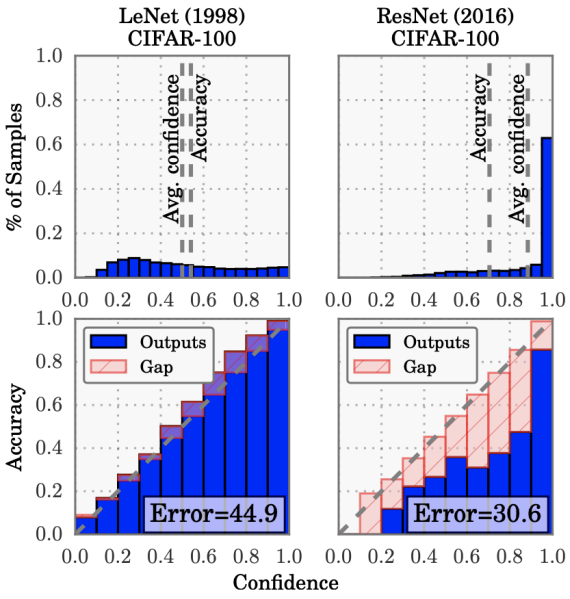


Figure 1: Confidence histograms (top) and reliability diagrams (bottom) for a 5-layer LeNet (left) and a 110-layer ResNet (right) on CIFAR-100.

iterating every element in B_m, then matching the true label y with label prediction $\hat{y}$, then take average over them.

$$\text{acc}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \mathbb{I}(\hat{y}_i = y_i) \tag{2}$$

We can also calculate average confidence over B_m by summing up prediction p_i every B_m, then take average over them.

$$\text{conf}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \hat{p}_i \tag{3}$$

Perfectly calibrated model should have acc = conf over every B_m. Therefore, we can observe from figure 1 that perfectly calibrated model should have identity curve.

ECE (Expected Calibration Error) is a scalar summary statistic of calibration. It is a weighted average of the difference between model accuracy and confidence across M bins and n samples. We have $\frac{|B_m|}{n}$ to ensure equal contribution to ECE over each bins Naeini et al. [2015]

$$\text{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{n} |\text{acc}(B_m) - \text{conf}(B_m)| \tag{4}$$

$$\mathbb{E}_{\hat{P}} \left[\left| \mathbb{P}\left(\hat{Y} = Y \mid \hat{P} = p\right) - p \right| \right] \tag{5}$$

MCE (Maximum Calibration Error) measures the worst-case deviation between accuracy and confidence. It is particularly important in high-risk applications where reliable confidence measures are crucial.Naeini et al. [2015]

$$\text{MCE} = \max_{m \in \{1,\ldots,M\}} |\text{acc}(B_m) - \text{conf}(B_m)| \tag{6}$$

$$\max_{p \in [0,1]} \left| \mathbb{P}\left(\hat{Y} = Y \mid \hat{P} = p\right) - p \right| \tag{7}$$

Problem with ECE There are some problem associated with ECE. First problem is discontinuity. By following the definition of ECE, we are computing expectation of difference between accuracy and confidence, but even if we have small discrepancy on the perfect forecaster, it result into large gap of ECE. For example, if we have uniform distribution X= $[a, b]$ (label a=0 and b=1), if we have perfect calibrator f=0.5, then we have ECE =0. However, if we have small perturbation $f(a) = 0.5-\epsilon$ and $f(b) = 0.5+\epsilon$, we have $ECE = (1/2)-\epsilon$, thus big gap just by small perturbation.

$$\left| \mathbb{P}(\hat{Y} = a \mid \hat{P} = 0.5 - \epsilon) - (0.5 - \epsilon) \right| = |0 - (0.5 - \epsilon)| = 0.5 - \epsilon$$

$$\left| \mathbb{P}(\hat{Y} = b \mid \hat{P} = 0.5 + \epsilon) - (0.5 + \epsilon) \right| = |1 - (0.5 + \epsilon)| = 0.5 - \epsilon$$

$$\text{ECE}(f_\epsilon) = \frac{1}{2}\left((0.5 - \epsilon) + (0.5 - \epsilon)\right) = 0.5 - \epsilon$$

Therefore, people often tried to solve this discontinuous problem by binned ECE method, which discretizing range to estimate the ECE However, this discretization turns out to matter in theory Kumar et al. [2019]. The ECE highly depends on number of bins M, and ECE become either 0 or $1/2$ depending on whether M is odd and even Kumar et al. [2019].

1.2.3 What determine good calibration metric

True calibration metric Błasiok et al. [2023] The f is predictor, D is distribution (x, y). The true distance to the calibration is distance between $f(x)$ (predictor) and $g(x)$ (perfectly calibrated predictor).The problem here is that it is assuming that we have fully access to entire domain X, which is impractical. In practical scenario, we often have access to $(f(x), y) \in D_f$ instead.

$$\text{dCE}_D(f) := \inf_{g \in \text{cal}(D)} \mathbb{E}_D |f(x) - g(x)|, \tag{8}$$

Sample Access (SA). In SA model, model depends on full domain X, which have access to $(x, f(x), y)$. (impractical)

Prediction-only access (PA). In the PA model, the model depends on D_f, which includes $(f(x), y)$. (practical)

Therefore, recent article proposed new metric depends on PA model Błasiok and Nakkiran [2023]

$$d((f_1, y_1), (f_2, y_2)) := \begin{cases} |f_1 - f_2| & \text{if } y_1 = y_2 \\ \infty & \text{otherwise} \end{cases} \tag{9}$$

$$\underline{\text{dCE}}(D) = \inf_{g \in \text{cal}(\mathcal{D})} W_1(f, g). \tag{10}$$

Requirements for Good calibration metric

Need to satisfy Robust completeness and soundness In the property testing framework, it is common to use definitions of completeness and soundness to decide whether notion has certain property Goldreich et al. [1998]. We use these properties to check whether it belongs to true calibration metric.

Completeness We want to have small value of measure when we have perfect calibrator.

$$\mu_D(f) = 0 \text{ if } f \in \text{cal}(\mathcal{D}) \tag{11}$$

Soundness We want to have large value of measure when model is not calibrated.

$$\mu_D(f) > 0 \text{ if } f \notin \text{cal}(\mathcal{D}) \tag{12}$$

However, these completeness and soundness do not give us specific value other than threshold value 0. Therefore, author defined new notions to provide specific value to satisfy completeness and soundness where a_1, a_2, c_1, c_2 are positive constants. For ideal calibration metric, we need to satisfy $a_1/a_2 \geq 2$. Błasiok and Nakkiran [2023]

$$\textbf{Robust Completness}: \quad \mu(D) \leq c_2 \underline{\text{dCE}}(D)^{\alpha_2} \tag{13}$$

$$\textbf{Robust Soundness}: \quad c_1 \underline{\text{dCE}}(D)^{\alpha_1} \leq \mu(D) \tag{14}$$

Therefore, metric should be in range as below equation.

$$c_1 \text{dCE}(D)^{\alpha_1} \leq \mu(D) \leq c_2 \text{dCE}(D)^{\alpha_2} \tag{15}$$

Need to be defined in PA model In practical machine learning senario, we usually don't have access to all feature vectors (X). For example, in patient disorder prediction task, there might be some hidden factors that contribute into patient medical condition such as life style. Therefore, we should assume our model is PA model.

Need to be continuous Major problem associated with previous metric was that it was discontinuous. As we mentioned above as two points example, even with small noise in the output, our ECE jumped from 0 to $0.5 - \epsilon$. Therefore, it is important to ensure metric is continuous.

1.2.4 Smooth ECE Błasiok and Nakkiran [2023]

Even though ECE has severe problem such as discontinuity, it is still being used in recent articles. Major reason is that ECE can be easily visualized into reliability diagram via integration of ECE as shown on figure 1.

Smooth ECE resolves problems associated with previous metric including both discontinuity and transition to reliability diagram.

Smooth ECE It leverages kernel density estimation using reflected Gaussian kernel in order to mitigate the bias introduced by standard Gaussian kernel near boundaries of the interval. We define smECE with using residual kernel smoothing term $\hat{r}_{D,}$ and kernel density estimation $\hat{\delta}_{D,K}$.

$$\tilde{K}_\sigma(x,y) := \sum_{\tilde{x}\in\pi_R^{-1}(x)} \phi_\sigma(\tilde{x}-y) = \sum_{\tilde{y}\in\pi_R^{-1}(y)} \phi_\sigma(x-\tilde{y}),$$

$$\hat{r}_{D,K}(t) := \frac{\mathbb{E}_{(f,y)\sim D}\left[K(t,f)(y-f)\right]}{\mathbb{E}_{(f,y)\sim D}\left[K(t,f)\right]}, \tag{16}$$

$$\hat{\delta}_{D,K}(t) := \mathbb{E}_{(f,y)\sim D}\left[K(t,f)\right], \tag{17}$$

$$\mathrm{smECE}_K(D) := \int \left|\hat{r}_{D,K}(t)\right| \hat{\delta}_{D,K}(t)\, dt, \tag{18}$$

$$\mathrm{smECE}_K(D) = \int \left|\mathbb{E}_{(f,y)\sim D}\left[K(t,f)(y-f)\right]\right| dt, \tag{19}$$

$$\tag{20}$$

Moreover smECE satisfy robust soundness and completeness.

$$\frac{1}{2}\underline{\mathrm{dCE}}(D) \le \mathrm{smECE}_h^*(D) \le 2\sqrt{\underline{\mathrm{dCE}}(D)}, \tag{21}$$

It can be easily visualized into realibility diagram as shown on figure 3.

1.3 What determines good calibration function

1.3.1 Calibration function should be trained on strictly proper scoring ruleBlasiok et al. [2024]

Scoring rule is a evaluation of predicted probability (Q) with respect to true probability (P) defined as $S(Q) \in R$. Strictly proper scoring rule is strong notion of scoring rule that expectation of true probability should be always greater than predicted probability.

$$E_p[S(p)] > E_p[S(q)] \quad \text{for} \quad q \neq p.$$

There are some example of strictly proper scoring rule used in machine learning domain such as Brier score (MSE) and Log score (Cross Entropy).

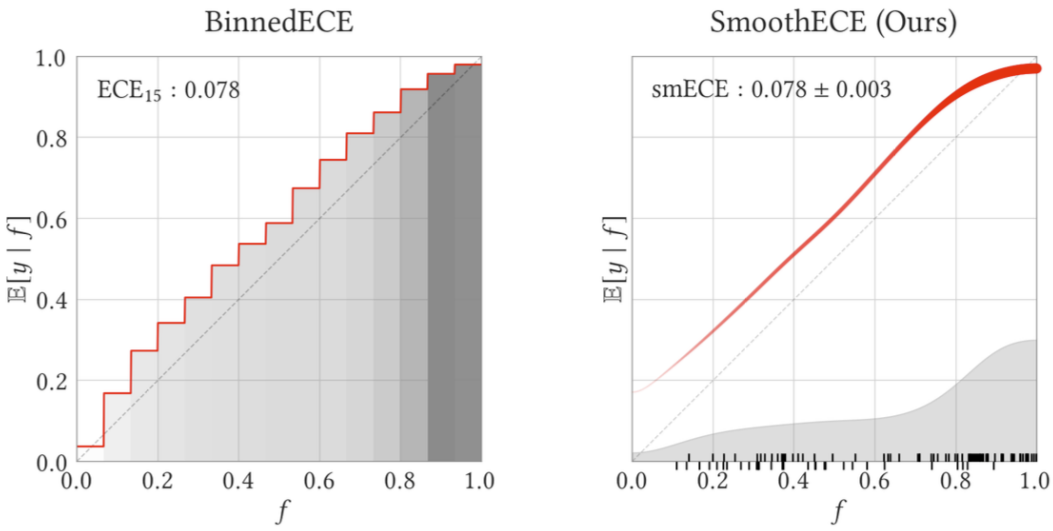


Figure 3: Confidence calibration of ResNet34 on ImageNet. Data from Hollemans (2020).

$$S_j(q) = 2q_j - \sum_{i \in I} q_i^2, \tag{22}$$

$$S_j(q) = \log q_j, \tag{23}$$

$$\tag{24}$$

We can simply prove that log scoring rule is indead strictly proper scoring rule using definition of KL divergence.

$$D_{KL}(p \| q) = \mathbb{E}_p \left[\log \frac{p(y \mid \mathbf{x})}{q_\theta(y \mid \mathbf{x})} \right] \geq 0$$

$$\mathbb{E}_p \left[\log p(y \mid \mathbf{x}) \right] \geq \mathbb{E}_p \left[\log q_\theta(y \mid \mathbf{x}) \right]$$

1.3.2 The calibration function should be strictly monotonic

When model is perfectly calibrated, reliability diagram has strictly monotonic property, so the calibration function should be naturally monotonic function. Moreover, strictly monotonic function preserves ranking of prediction, thus not affecting accuracy.

1.3.3 The calibration function should be flexible

Miscalibration may not fit a specific parametric form so we need a non-parametric model.

1.3.4 The calibration function needs to be trained on independent data

We should use validation data to fit calibration function

1.4 Calibration in Classification

Post-hoc calibration methods calibrate the model during the validation process.

1.4.1 Canonical Calibration

It is a strict definition of calibration, which it ensures calibration on every individual instances. $q_{Y|X}$ is the predicted distribution (cdf) and $q_{Y|X}(y)$ is the probability evaluated at y.

$$\mathbb{P}(Y = y \mid q_{Y|X}) = q_{Y|X}(y) \tag{25}$$

It can be expressed as distribution matching between true Y and predicted $\hat{Y}$ given model cdf $q_Y|X$.

$$Y = \hat{Y} \mid q_{Y|X} \tag{26}$$

1.4.2 Top-Label Calibration

It is weaker condition of calibration where it only guarantees calibration on top label. It is used on multiclass classification where it focuses on $Y^* := \arg\max_{y \in Y} q_{Y|X}(y)$.

$$\mathbb{P}(Y = Y^* \mid q_{Y|X}(Y^*)) = q_{Y|X}(Y^*) \tag{27}$$

It can be written as distribution matching between specific class of Y and $\hat{Y}$ given probability of $q_{Y|X}(Y^*)$

$$1\{Y = Y^*\} \triangleq 1\{Y = Y^*\} \mid q_{Y|X}(Y^*) \tag{28}$$

Marginal Calibration It focuses calibration on overall class of Y instead of every individual instances as in canonical calibration.

$$\mathbb{P}(Y = y \mid q_{Y|X}(y)) = q_{Y|X}(y) \tag{29}$$

It can be written as distribution matching of overall class of y within Y and $\hat{Y}$ given marginal probability $q_{Y|X}(y)$ for all $y \in Y$.

$$1\{Y = y\} \triangleq 1\{Y = y\} \mid q_{Y|X}(y) \tag{30}$$

1.4.3 Calibrating Binary Models

We first examine binary case, where model outputs confidence for positive for convenience to avoid repetition of considering negative class. i.e.

$$h(X) = (\hat{Y}, \hat{P}) \text{ where } \{\hat{Y} == 1 : \hat{Y} = \{0, 1\} \ \& \ \hat{P} = \sigma(Z)\}$$

Our model produces both logits z_i and label y_i. We use sigmoid function to generate $\sigma(z_i) = p_i$.

Histogram binning (Non-Parametric)Zadrozny and Elkan [2001] Cons: Highly dependent on bin size

Algorithm 1 Original Histogram Binning Algorithm

1: **Input:** Model outputs $h(X) = (\hat{Y}, \hat{P})$ from validation data set D_n, number of bins M
2: **Output:** Calibrated probability q_i
3: predicted probability (confidence score) $\hat{p}_i$ is divided into M bins $B_1, ..., B_M$
4: Each bins are assigned with calibrated score

$$\theta_m = \frac{1}{|B_m|} \sum_{\hat{y}_i \in B_m} \hat{y}_i$$

5: **for** $\hat{y}_i, \hat{p}_i$ in $\{\hat{Y}, \hat{P}\}$ **do**
6: assign $\hat{p}_i$ into B_M
7: set $\hat{q}_i = \theta_m$
8: **end for**
9: **Return:** Calibrated probability Q

It is trying to minimize MSE between prediction label y_i and calibration score $theta_m$ thus following strictly proper scoring rule.

$$\min_{\theta_1, ..., \theta_M} \sum_{m=1}^{M} \sum_{i=1}^{n} \mathbf{1}(a_m \leq \hat{p}_i < a_{m+1})(\theta_m - \hat{y}_i)^2 \tag{31}$$

Isotonic Regression (Non-Parametric) Zadrozny and Elkan [2002]

The isotonic regression learns piecewise function that calibrate the $\hat{p}_i$; ie. $\hat{q}_i = f(\hat{p}_i)$. It has constraint that interval boundaries $a_1, ..., a_M + 1$ are monotonically increasing, so it is suitable in calibration since calibration plot ideally monotonically increasing. Common way to solve this isotonic regression is pair-adjacent violators (PAV)Zadrozny and Elkan [2002], which finds step wise function that reduces MSE error. The cons of isotonic regression is that it is prone to over fitting.

$$\begin{aligned} &\min_{\substack{M \\ \theta_1, ..., \theta_M \\ a_1, ..., a_{M+1}}} \sum_{m=1}^{M} \sum_{i=1}^{n} 1(a_m \leq \hat{p}_i < a_{m+1})(\theta_m - y_i)^2 \\ &\text{subject to} \quad 0 = a_1 \leq a_2 \leq \cdots \leq a_{M+1} = 1, \\ &\qquad \theta_1 \leq \theta_2 \leq \cdots \leq \theta_M. \end{aligned} \tag{32}$$

Algorithm 2 PAV algorithm for estimating posterior probabilities from uncalibrated model predictions.

Input: training set (f_i, y_i) sorted according to f_i
Initialize $\hat{m}_{i,i} = y_i$, $w_{i,i} = 1$
while $\exists i$ s.t. $\hat{m}_{k,i-1} \geq \hat{m}_{i,l}$ **do**
    Set $w_{k,l} = w_{k,i-1} + w_{i,l}$
    Set $\hat{m}_{k,l} = \frac{w_{k,i-1}\hat{m}_{k,i-1} + w_{i,l}\hat{m}_{i,l}}{w_{k,l}}$
    Replace $\hat{m}_{k,i-1}$ and $\hat{m}_{i,l}$ with $\hat{m}_{k,l}$
end while
Output the stepwise constant function:

$$\hat{m}(f) = \hat{m}_{i,j}, \text{ for } f_i < f \leq f_j$$

Platt scaling (Parametric) In the binary scheme, it leverages Sigmoid function to obtain calibrated probability using model output. It is parametric approach which has one parameter called Temperature. We optimize this temperature minimizing log cross entropy loss using validation set while freezing original model parameters.

$$\hat{q}_i = \frac{1}{1 + \exp(T * \hat{p}_i)} \tag{33}$$

$$\text{argmin}_T \left\{ -\sum_i \left[y_i \log(\hat{p}_i) + (1 - y_i) \log(1 - \hat{p}_i) \right] \right\} \tag{34}$$

Beta Calibration The problem with platt scaling is that it only maps to the sigmoid function. The main problem associated with it is that derivation of logistic calibration assumes normal distribution of dataset, which requires infinite support. Therefore, it is practical to use distribution with finite support such as Beta distribution Kull et al. [2017]. Moreover, Beta Calibration can map to the identity function, which is useful when we already have calibrated probability. We fit the parameter minimizing log loss, which is considered as strictly proper scoring rule.

Algorithm 3 Beta[a=b] calibration via logistic regression

Input: $\mathbf{y}$ and $\hat{\mathbf{p}}$ are the label and model output score vectors on validation instances, $\hat{\mathbf{p}}$ is the model output score vector on validation instances

1: $\hat{\mathbf{p}}' \leftarrow \ln \frac{\hat{\mathbf{p}}}{1-\hat{\mathbf{p}}}$

2: $(a, c) \leftarrow$ fit univariate logistic regression to predict $\mathbf{y}$ from $\hat{\mathbf{p}}'$

3: $\hat{q}_i \leftarrow 1/\left(1 + 1/\left(e^{c\left(\frac{\hat{\mathbf{p}}}{1-\hat{\mathbf{p}}}\right)^a}\right)\right)$

4: **return** $\hat{q}_i$

1.4.4 Multi-class Models

We extend this to multi-class case where number of classes are $K > 2$. In this case our model outputs logits (Z) as vector, and applying softmax to obtain probability $\sigma_{sm}(Z) = \hat{P}$. Finally, our model outputs both probability ($\hat{P}$) and predicted class ($\hat{Y}$) by taking argmax of logits (Z) as follows $h(X) = (\hat{P}, \hat{Y})$. Our goal is to obtain calibrated probability q_i leveraging $\hat{P}, \hat{Y}, and\ Z$.

$$\sigma_{\mathrm{SM}}(\mathbf{z}_i)^{(k)} = \frac{\exp(z_i^{(k)})}{\sum_{j=1}^{K} \exp(z_i^{(j)})}$$

Extension of binning methods We can simply extend binary calibration method to multi class case by turning into one verses all problem. We use binary label to indicate each K class. We obtain $\left[\hat{q}_i^{(1)}, .., \hat{q}_i^{(K)}\right]$ un-normalized calibrated vectors using K number of models for each class. We then take the argmax of un-normalized calibrated vector to obtain $\hat{y'}_i$ and $\hat{q'}_i$ by normalizing probability of calibrated vector. This can be applied to binning and isotonic regression.

Extension of Platt scaling

Matrix scaling This is simple extension to Platt scaling where now z_i is the vector instead of single value. We optimize parameters W and b by minimzing NLL loss using validation set.

$$\hat{q}_i = \max_k \sigma_{\mathrm{SM}}(\mathbf{W}\mathbf{z}_i + \mathbf{b})^{(k)},$$
$$\hat{y}_i' = \arg\max_k (\mathbf{W}\mathbf{z}_i + \mathbf{b})^{(k)}.$$

Temperature Scaling This method is most common calibration method used in recent research. Instead of applying linear transformation, we just normalize logit z_i with single parameter T temperature. We again optimize this T respect to NLL loss. This has many benefits over matrix

scaling. First, we only have one paramter. Second, it doesn't affect the model's accuracy since scaling by T doesn't change the maximum value of calibrated probability q_i.

$$\hat{q}_i = \max_k \sigma_{\text{SM}} \left(\frac{\mathbf{z}_i}{T} \right)^{(k)}. \tag{35}$$

1.4.5 Evaluation of Calibration Methods

Table 1: Comparison of Calibration Methods

	Calibration Methods			
	Histogram Binning	**Isotonic Regression**	**Temperature Scaling**	**Beta Calibration**
1. Strictly Proper Score	MSE	MSE	Log	Log
2. Monotonic Increasing	×	◯	◯	◯
3. Using Validation Dataset	◯	◯	◯	◯
4. Non-parametric	◯	◯	×	×

1.5 Calibration in Regression

1.5.1 Quantile Calibration Kuleshov et al. [2018]

Quantile calibration is baseline algorithm for calibration in regression. The definition of Quantile Calibration is same as cdf of Uniform(0,1) by probability integral transform. Therefore, it is matching predicted conditional cdf $Q_{Y|X}$ to true conditioned cdf of uniform $P_{Y|X}$.

$$P(Q_{Y|X}(Y) \leq t) = t, \quad \forall t \in [0,1] \tag{36}$$

$$Q_{Y|X}(Y) = P_{Y|X}(Y) \tag{37}$$

Algorithm 4 Quantile Calibration

Input: Uncalibrated model $H : \mathcal{X} \to (\mathcal{Y} \to [0,1])$ and calibration set $S = \{(x_t, y_t)\}_{t=1}^{T}$
Output: Auxiliary recalibration model $R : [0,1] \to [0,1]$

1: Construct a recalibration dataset:

$$\mathcal{D} = \left\{ \left([H(x_t)](y_t), \hat{P}\left([H(x_t)](y_t) \right) \right) \right\}_{t=1}^{T},$$

$$\text{where} \quad \hat{P}(p) = \frac{|\{y_t \mid [H(x_t)](y_t) \leq p, t = 1, \ldots, T\}|}{T}$$

2: Train a model R (e.g., isotonic regression) on $\mathcal{D}$.

1.5.2 Threshold Calibration Sahoo et al. [2021]

The distribution calibration theoretically guaranteed to yield zero realiability gap, but it is hard to achieve in practice since training $p(y|x)$ requires multiple x, but in practice, we only have access to one x Song et al. [2019]. Moreover, flexible distribution families can better approximate the true

distribution than simple one, which was approximated by distribution calibration, thus result in lower loss.

Bayes decision Rule A decision rule $\delta : X- > A$, which maps feature to Action. Loss function uses X, Y, A to compute the loss. Since true Y is usually unobserved, it is practical to use model output label $\tilde{Y} \sim h[X]$. We can obtain optimal decision δ_h^x. **Threshold calibration is only considering binary action space** $A \in [0, 1]$, and decision loss is summation of decision cost c with taking account of decision threshold y_0.

$$\delta_h^* = \arg\inf_{\delta \in \Delta} \mathbb{E}_X \mathbb{E}_{\tilde{Y} \sim h[X]} \left[\ell(X, \tilde{Y}, \delta(X)) \right] \tag{38}$$

$$\ell(x, y, a) = \sum_{i \in \{0,1\}} c_{1,i} \mathbb{I}(y \leq y_0, a = i) + \sum_{i \in \{0,1\}} c_{0,i} \mathbb{I}(y > y_0, a = i), \tag{39}$$

We define **threshold decision rule** as follows. We are taking action whether decision threshold $h[x](y_0)$ is either greater or less then threshold parameter $a \in [0, 1]$. In here, $h[x]$ is cdf given feature X, and $h[x][y_0]$ is probability given at point y_0.

$$\delta_h^*(x) = \mathbb{I}(h[x](y_0) \geq \alpha) \text{ or } \delta_h^*(x) = \mathbb{I}(h[x](y_0) \leq \alpha) \tag{40}$$

Reliability Gap It is a metric being used to quantify accuracy of decision loss l. However, zero reliability gap does not guarantee each individual decision is optimal since reliability gap is average decision loss.

Threshold Calibration Intuitively, threshold calibration ensures that the forecaster satisfies average calibration ($P[h[X](Y) < c] = c$) on the subsets of predicted CDFs ($h[X](y_0)$)where the decision maker chooses a = 0 and a = 1. This is equvalent as matching predicted conditional cdf $Q_{Y|X}$ to true conditioned cdf $P_{Y|X}$ with given threshold condition.

$$P[Q_{Y|X}(Y) \leq t \mid Q_{Y|X}(y_0) \leq \alpha] = t \tag{41}$$

$$Q_{Y|X}(Y) = P_{Y|X}(Y) \mid Q_{Y|X}(y_0) \leq \alpha \tag{42}$$

Threshold Calibration Error It measures deviation of cost between true cost c with predicted cost terms that are coming from taking binary action where $h[X](y_0) \leq \alpha$ and $\mid h[X](y_0) > \alpha$

$$\begin{aligned} TCE(h, y_0, \alpha) = &\int_0^1 |\Pr[h[X](Y) \leq c \mid h[X](y_0) \leq \alpha] - c| \, dc \\ &+ \int_0^1 |\Pr[h[X](Y) \leq c \mid h[X](y_0) > \alpha] - c| \, dc. \end{aligned} \tag{43}$$

Example Lets consider following example to better understand threshold calibration. We have ML model that predict patient's length of stay (LOS) in hospital to decide whether to admit new patient. Decision rule is that if LOS is less than threshold day y_0, we can admit new patients.

If we want to consider case where threshold day is 3 with probability of less or equal to 0.5, i.e. $h[X][3] \leq 0.5$. We can check whether it satisfies threshold calibration using below equation.

$$\Pr[h[X](Y) \leq c \mid h[X](3) \leq 0.5] = c \tag{44}$$

Relationship with other method If model is distribution calibrated, then it is threshold calibrated. Also, threshold calibrated model satisfies quantile calibration. However, the reverse statement is not true.

Algorithm At each step of itteration, we find y_0 and α that maximize threshold calibration error. Next, we partition into two sub groups using above values. Finally we apply isotonoic regression on each group to optain threshold calibrated model.

Algorithm 5 Threshold Recalibration

Input: Forecaster $h : \mathcal{X} \to \mathcal{F}(\mathcal{Y})$, maximum error $\epsilon > 0$
Output: A threshold-calibrated forecaster

Set $h^0 \leftarrow h$
for $t = 1, 2, \ldots$ **until** maximum threshold calibration error $\sup_{y_0,\alpha} \mathrm{TCE}(h^{t-1}, y_0, \alpha) \leq \epsilon$ **do do**
    Find the y_0 and α that maximize threshold calibration error.
    $y_0^t, \alpha^t \leftarrow \arg\sup_{(y_0,\alpha)\in\mathcal{Y}\times[0,1]} \mathrm{TCE}(h^{t-1}, y_0, \alpha)$
    Partition input features $\mathcal{X}$ into $\mathcal{X}_0 \leftarrow \{x \in \mathcal{X} \mid h^{t-1}[x][y_0^t] \leq \alpha^t\}$ and $\mathcal{X}_1 = \mathcal{X} \setminus \mathcal{X}_0$.
    Use Isotonic regression to learn recalibration maps $\phi_0^t, \phi_1^t : \mathcal{F}(\mathcal{Y}) \to \mathcal{F}(\mathcal{Y})$ on $\mathcal{X}_0$ and $\mathcal{X}_1$ respectively.
    Apply the recalibration map to obtain new prediction functions.
    $h^t[x] \leftarrow \begin{cases} \phi_0^t(h^{t-1}[x]) & \text{if } x \in \mathcal{X}_0 \\ \phi_1^t(h^{t-1}[x]) & \text{otherwise} \end{cases}$
end for
return h^T where T is the final iteration count.

1.5.3 Marginal Calibration Gneiting and Katzfuss [2014]

We say model is marginal calibrated if expectation of model cdf matches true marginal probability. This is matching $\hat{Y} \sim E[Q_{Y|X}]$ to $Y \sim P_Y$. Thus $Y = \hat{Y}$.

$$E[Q_{Y|X}(y)] = P_Y(y) \text{ for all y} \tag{45}$$

1.5.4 distribution calibration Song et al. [2019]

The problem with quantile calibration is that it only depends on marginal probability, which only guarantee to be calibrated on average over all prediction. However, distribution calibration ensures that entire predicted distribution match the true distribution for each individual prediction. We can formalize this as shown on equation 13 where Q_0 is all possible all possible prediction of Q. Distribution calibration is distribution matching by matching model prediction $\hat{Y}$ to true Y given condition of model prediction $Q_{Y|X}$.

$$P(Y \leq y \mid Q_{Y|X} = Q_0) = Q_0(y) \tag{46}$$

$$P_{Y|X}(y) \mid Q_{Y|X} = Q_{Y|X}(y) \text{ for all } y \in \mathcal{Y} \tag{47}$$

$$\hat{Y} = Y \mid Q_{Y|X} \tag{48}$$

We can observe that why post hoc calibration help improving the calibration metric using NLL example. Below is decomposition of log proper scoring rule into calibration term and sharpness term. The sharpness term is representing the distribution $(\log p(y|s_x)))$,which become fixed after training since distribution is based on parameters and parameters are fixed. Therefore, post-hoc calibration doesn't affect the sharpness **if and only if we use non parametric post hoc calibration.**

$$\mathbb{E}_{(X,Y)}\left[-\ln Q_{Y|X}(y)\right] = \int p(x,y)\left[-\ln Q_{Y|X}(y)\right]\, dx\, dy \tag{49}$$

$$\mathbb{E}_{(X,Y)}\left[-\ln Q_{Y|X}(y)\right] = \int p(x)p(y \mid Q_{Y|X})\left[-\ln Q_{Y|X}(y)\right]\, dx\, dy \tag{50}$$

$$\ln \frac{p(y \mid Q_{Y|X})}{Q_{Y|X}(y)} = \ln p(y \mid Q_{Y|X}) - \ln Q_{Y|X}(y) \tag{51}$$

$$\mathbb{E}_{(X,Y)}\left[-\ln Q_{Y|X}(y)\right] = \int p(x)p(y \mid Q_{Y|X})\left[\ln \frac{p(y \mid Q_{Y|X})}{Q_{Y|X}(y)} - \ln p(y \mid Q_{Y|X})\right] dx\, dy \tag{52}$$

$$\begin{aligned}\mathbb{E}_{(X,Y)}\left[-\ln Q_{Y|X}(y)\right] = &\int p(x)\left[\int p(y \mid Q_{Y|X})\left[\ln \frac{p(y \mid Q_{Y|X})}{Q_{Y|X}(y)}\right] dy\right] dx \\ &+ \int p(x)\left[\int p(y \mid Q_{Y|X})\left[-\ln p(y \mid Q_{Y|X})\right] dy\right] dx\end{aligned} \tag{53}$$

$$\int p(y \mid Q_{Y|X})\left[-\ln \frac{p(y \mid Q_{Y|X})}{Q_{Y|X}(y)}\right] dy = \mathrm{KL}(p(y \mid Q_{Y|X}) \| Q_{Y|X}(y)) \tag{54}$$

$$\int p(x)p(y \mid Q_{Y|X})\left[\ln p(y \mid Q_{Y|X})\right]\, dy\, dx = \mathbb{E}_{(X,Y)}\left[-\ln p(y \mid Q_{Y|X})\right] \tag{55}$$

$$\begin{aligned}\mathbb{E}_{(X,Y)}\left[-\ln Q_{Y|X}(y)\right] = &\mathbb{E}_X\left[\mathrm{KL}(p(y \mid Q_{Y|X}) \| Q_{Y|X}(y))\right] \\ &+ \mathbb{E}_{(X,Y)}\left[-\ln p(y \mid Q_{Y|X})\right]\end{aligned} \tag{56}$$

algorithm for distribution calibration It leverages Beta calibration maps to transform CDF of distribution output of regression (u, σ). This paper utilizes GP to learn parameter of Beta Calibration since during the training we can only observe x_i once, so it is not enough to train $Q_{Y|X}$, which requires multiple samples of inputs. Therefore we need to use GP to obtain output that are close to predicted (u, σ).

1.5.5 Decision Calibration

Definition We consider model is L^K decision calibrated when simulated loss(left term) and true loss (right term) matches where $L^K = \{\ell : Y \times A \to \mathbb{R} \mid |A| = K\}$. This is equivalent as matching

$Y = \hat{Y}$.

$$\mathbb{E}_X \mathbb{E}_{\hat{Y} \sim Q_{Y|X}} [\ell(\hat{Y}, \delta(Q_{Y|X}))] = \mathbb{E}_X \mathbb{E}_{Y \sim P_{Y|X}} [\ell(Y, \delta(P_{Y|X}))] \tag{57}$$

There are two properties that we can obtain when model is decision calibrated.

No regret is that if we choose optimal decision δ^*, then it should have lowest loss among other decision rule δ. Intuitively, we cannot choose any better action if we followed optimal decision rule under true loss function.

Accurate loss estimation is that under optimal decision rule, our loss is equal when label is either drawn from model ($\hat{Y}$) or true distribution (Y). This is important since true Y are revealed with significant delay or never revealed. For example, for financial decision task, the agents may need to observe long time of period of market condition before making decision. Therefore, this property help us to trust our model prediction, thus we do not need to wait until true outcome revealed.

$$\mathbb{E}_X \left[\mathbb{E}_{Y \sim P_{Y|X}} \left[\ell(Y, \delta^*(Q_{Y|X})) \right] \right] \leq \mathbb{E}_X \left[\mathbb{E}_{Y \sim P_{Y|X}} \left[\ell(Y, \delta(Q_{Y|X})) \right] \right] \quad \text{(No regret)}$$

$$\mathbb{E}_X \left[\mathbb{E}_{\hat{Y} \sim Q_{Y|X}} \left[\ell(\hat{Y}, \delta^*(Q_{Y|X})) \right] \right] = \mathbb{E}_X \left[\mathbb{E}_{Y \sim P_{Y|X}} \left[\ell(Y, \delta^*(Q_{Y|X})) \right] \right] \quad \text{(Accurate loss estimation)}$$

One important note here is that decision calibration doesn't guarantee individual decision since it only looks at average loss. Moreover, decision calibration doesn't have decision rule that directly depend on X since it requires multi-calibration.

Connection to distribution calibration Distribution calibration has attractive property in theoretical perspective that it perfectly matches target and true distribution. However, it is hard to achieve in practice due to difficulty training $H_{Y|X}$ with limited sample of X. In practical setting, we have bounded action space K. Therefore, if model is $L^K decisioncalibrated$, we can ensure that model is distribution calibrated with constraint of K space.

Approximated definition It is hard to achieve formal definition of decision calibration as shown on equation 32. Therefore, we introduce approximated definition. First, we need to introduce new notation B^K, which is linear multi-class classification function where it inputs probability distribution q with weight vector associated with action w_a. It takes highest vector to decide optimal action.

$$B^K = \left\{ b_w \mid \forall w \in \mathbb{R}^{K \times C} \right\} \quad \text{where} \quad b_w(q, a) = \mathbb{I} \left(a = \arg \sup_{a' \in [K]} \langle q, w_{a'} \rangle \right) \tag{58}$$

Model is L^k decision calibrated with error ϵ if it satisfies below equation. Intuitively, it is taking suprema of discrepancy between prediction and true outcome when it matches optimal action.

$$\sup_{b \in B^K} \sum_{a=1}^{K} \left\| \mathbb{E} \left[(\hat{Y} - Y) b(h(X), a) \right] \right\|_2 \leq \epsilon \tag{59}$$

Algorithm This algorithm is trying to find the worst case violation of equation 34, then try to adjust the model using adjustment found by d_a.

Algorithm 6 Recalibration algorithm to achieve $\mathcal{L}^K$ decision calibration.

Input: current prediction function $\hat{h}$, tolerance ϵ. Initialize $\hat{h}^{(0)} = \hat{h}$;
for $t = 1, 2, \cdots, T$ **until** output $\hat{h}^{(T)}$ when it satisfies Eq. (8) **do do**
    Find $b \in B^K$ that maximizes $\sum_{a=1}^{K} \left\| \mathbb{E}\left[(Y - \hat{h}^{(t-1)}(X)) b(\hat{h}^{(t-1)}(X), a) \right] \right\|$;
    Compute the adjustments $d_a = \frac{\mathbb{E}[(Y - \hat{h}^{(t-1)}(X)) b(\hat{h}^{(t-1)}(X), a)]}{\mathbb{E}[b(\hat{h}^{(t-1)}(X), a)]}$;
    Set $\hat{h}^{(t)} : x \mapsto \hat{h}^{(t-1)}(x) + \sum_{a=1}^{K} b(\hat{h}^{(t-1)}(x), a) \cdot d_a$ (projecting onto $[0, 1]$ if necessary);
end for

1.5.6 Group Calibration Pleiss et al. [2017]

Equalized Odds Hardt et al. [2016]: The group calibration follows the Equalized Odds framework. The equalized odds assume two disjoint group exist G_1, G_2. We sample binary classifier $h_1 \sim G_1$ and $h_2 \sim G_2$. The main criteria is that it must have equal false positive and false negative rate ($\text{cfp}(h_1) = \text{cfp}(h_2)$ and $\text{cfn}(h_1) = \text{cfn}(h_2)$).

It is commonly accepted amongst practitioners that both h_1 and h_2 are calibrated, but it is hard to satisfy both calibration and equalized odd.Kleinberg et al. [2016]

$$\text{cfp}(h_t) = \mathbb{E}_{(x,y)\sim G_t}[h_t(x)|y = 0] \tag{60}$$

$$\text{cfn}(h_t) = \mathbb{E}_{(x,y)\sim G_t}[(1 - h_t(x))|y = 1] \tag{61}$$

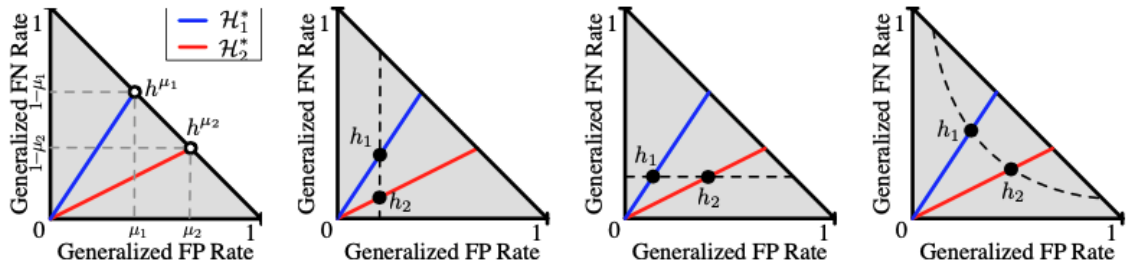
Geometric interpretation For trivial classifier which outputs $h(x) = c$, we have $c_{fp} = c$ and $c_{fn} = 1 - c$, thus $c_{fp}(h) + c_{fn}(h) = 1$. Therefore, to be better than the trivial classifier, our classifier must lies below than the diagonal line. We denote calibrated set of possible classifiers for group G_1 and G_2 as H_1^* and H_2^*, and we have following linear relationship as shown below equation, where u_t is base rate, which is probability of belongs to positive class.

$$\text{cfn}(h_t) = \frac{(1 - \mu_t)}{\mu_t} \text{cfp}(h_t) \tag{62}$$

Intuitively, h_1 lies on the slope $(1 - u_1)/u_1$ and h_2 lies on the slope $(1 - u_2)/u_2$ as shown on figure below. **In order to satisfy equalized odd, our model need to lie between same coordinate in the false positive and false negative plane as shown on figure b and c, which is hard to satisfy in practice since only point it can satisfies is origin (perfect predictor)** Moreover, better classifier lies close to the origin where it has lower false negative and positive rate.

Relaxed Equalized Odd condition Therefore, we need to relax this condition of Equalized odd to satisfy both calibration and equalized odd. We define cost function where a_t and b_t are non negative constants, which give us option to trade off of error rate between false positive and negative rate.

$$g_t(h_t) = a_t \text{cfp}(h_t) + b_t \text{cfn}(h_t) \tag{63}$$



(a) Possible cal. classifiers $\mathcal{H}_1^*, \mathcal{H}_2^*$ (blue/red). (b) Satisfying cal. and equal F.P. rates. (c) Satisfying cal. and equal F.N. rates. (d) Satisfying cal. and a general constraint.

Figure 1: Calibration, trivial classifiers, and equal-cost constraints – plotted in the false-pos./false-neg. plane. $\mathcal{H}_1^*, \mathcal{H}_2^*$ are the set of cal. classifiers for the two groups, and h^{μ_1}, h^{μ_2} are trivial classifiers.

In order to satisfy relaxed equalized odd we must have same cost function for h_1 and h_2.

$$g_1(h_1) = g_2(h_2) \tag{64}$$

Algorithm We assume that generally $g_1(h_1) \geq g_2(h_2)$, and we want to find $\tilde{h_2}$ that satisfies $g_1(h_1) = g_2(\tilde{h2})$. This can be found by $h_2(x)$ to output u_2 with some probability $\alpha = \frac{g_1(h_1)-g_2(h_2)}{g_2(h_{\mu_2})-g_2(h_2)}$ obtained by linear interpolation of h_2 and h^{u_2}(trivial classifier).

$$\tilde{h}_2(\mathbf{x}) = \begin{cases} h^{\mu_2}(\mathbf{x}) = \mu_2 & \text{with probability } \alpha \\ h_2(\mathbf{x}) & \text{with probability } 1-\alpha \end{cases} \tag{65}$$

Algorithm 7 Achieving Calibration and an Equal-Cost Constraint via Information Withholding

Input: classifiers h_1 and h_2 s.t. $g_2(h_2) \leq g_1(h_1) \leq g_2(h^{\mu_2})$, holdout set P_{valid}.

- Determine base rate μ_2 of G_2 (using P_{valid}) to produce trivial classifier h^{μ_2}.
- Construct $\tilde{h}_2$ using with $\alpha = \frac{g_1(h_1)-g_2(h_2)}{g_2(h^{\mu_2})-g_2(h_2)}$, where α is the interpolation parameter.

return h_1, $\tilde{h}_2$ — which are calibrated and satisfy $g_1(h_1) = g_2(\tilde{h}_2)$.

Cons If group is unknown due to privacy reason in loan service example, it is hard to apply this algorithm.

1.6 Local Calibration

Definition Local calibration groups similar feature sample into a soft neighborhood with neighborhood size parameter γ. It uses kernel function k_γ since similar sample shares similar calibration. ϕ is the feature map function $\phi : X \to R$ and g is the Lipschitz function. As γ increases, bandwidth of neighbor grows.

$$k_\gamma(x, x') = g\left(\frac{\phi(x) - \phi(x')}{\gamma}\right) \tag{66}$$

We define local calibration as below equation where $\hat{p} : X \to [0, 1]$ is confidence predictor and $h : X \to Y$ as classifier. It is prediction error weighted by kernel score, and we only consider case where confidence of random sample and anchor sample matches $\hat{p}(x') = \hat{p}(x)$. As $\gamma \to 0$, it becomes individual calibration. If $\gamma \to inf$, it becomes distribution calibration.

$$\sup_{x \in \text{supp}(P)} \left(\mathbb{E}_{(x',y') \sim P} \left[\left(\hat{p}(x') - 1[h(x') = y'] \right) \cdot k_\gamma(x, x') \mid \hat{p}(x') = \hat{p}(x) \right] \right) = 0. \tag{67}$$

We can compute local calibration error by letting $B(x)$ be the set of indices of the points in data occupying the same confidence acting like bins. Intuitively, it is difference between confidence and accuracy weighted by kernel score. Only difference between ECE is the kernel weight factor.

$$\text{LCE}_\gamma(x; h, \hat{p}) = \left| \frac{\sum_{i \in \beta(x)} (\hat{p}(x_i) - 1[h(x_i) = y_i]) k_\gamma(x, x_i)}{\sum_{i \in \beta(x)} k_\gamma(x, x_i)} \right|. \tag{68}$$

Algorithm We obtain post-hoc calibration method with computing confidence using below equation. If $\gamma- > 0$, it becomes nearest neighbor method, and $\gamma- > inf$ makes histogram binning method with kernel weighting factor.

$$\hat{p}'(x) = \frac{\sum_{i \in \beta(x)} k_\gamma(x, x_i) 1[h(x_i) = y_i]}{\sum_{i \in \beta(x)} k_\gamma(x, x_i)}. \tag{69}$$

1.7 Regularization Methods

The regularization methods applies calibration during the training process with adding regularization term to the training objective.

1.7.1 Individual Calibration(Regularization methodZhao et al. [2020]

Definition Since perfect calibrator has uniform distribution by using properties of inverse CDF theorem i.e. $F_{Y|x}(y) = y$. Our model $H[x](Y)$ should be uniformly distributed, so we define error term by measuring the distance between our model $F_{H[x](Y)}$ and uniform distribution F_U. If error term equals to zero, we say it is individually calibrated.

$$\text{err}_H(x) \stackrel{\text{def}}{=} d(F_{H[x](Y)}; F_U) \tag{70}$$

However, in practice it is hard to achieve above definition, so we instead use approximated definition as shown below.

$$P\left[\text{err}_H(X) \leq \epsilon\right] \geq 1 - \delta \tag{71}$$

We can express it as distribution matching term as follows. Since equation below is same as quantile calibration except including all of X, so it can be expressed as matching uniform (0,1) distribution $P_{Y|X}$ to $Q_{Y|X}$ conditioned on X.

$$\mathbb{P}\left(Q_{Y|x}(Y) \leq t\right) = t, \text{ for all } t \in [0, 1] \text{ and } x \in \mathcal{X} \tag{72}$$

$$Q_{Y|X}(Y) \triangleq P_{Y|X}(Y) \,|\, X = x \tag{73}$$

$$Q^{-1}_{Y|X}(Q_{Y|X}(Y)) \triangleq Q^{-1}_{Y|X}(P_{Y|X}(Y)) \,|\, X = x \tag{74}$$

$$Y \triangleq Q^{-1}_{Y|X}(U) \,|\, X = x \tag{75}$$

$$Y \triangleq \hat{Y} \,|\, X = x \tag{76}$$

Relation between Average Calibration (Quantile Calibration) The major difference between average calibration ($H[x](Y)$) and individual calibration ($H[X](Y)$) is that individual calibration guarantees calibration on every instances of x, but average calibration guarantees calibration on average of X, which may be problematic if X has some other distribution, ex. separation by gender.

$$\text{err}_H(x) \stackrel{\text{def}}{=} d(F_{H[X](Y)}; F_U) \tag{77}$$

Relation between Group Calibration Group calibration resolves the cons of average calibration by guaranteeing calibration on sub group $S_1, ..., S_k \in X$. However, it has strong requirement that group must be known.

$$\forall i \in [k];\ d\left(F_{H[X_{S_i}](Y)}, F_U\right) \leq \epsilon \tag{78}$$

Randomized Forecasting It is hard to achieve individual calibration in deterministic forecasters (Linear Regression & SVM) since if it is deterministic, forecaster distribution must be identical to the true distribution, which is impractical. Therefore, we need to use randomized forecaster, where $R \sim F_U$ is the random seed and $\bar{h}$ is randomized forecaster.

$$\mathbf{H}[x] = \bar{h}[x, R] \tag{79}$$

Original individual calibration requires $\mathbf{H}[x](Y)$ to be uniform, but it is hard to achieve since we only have access to one sample x. Therefore, we instead try to verify $\bar{h}[x, R](y)$ is uniform. This can be easily assumed if we assume that $\bar{h}$ is monotonic, then by following inverse CDF theorem, $\bar{h}[x, r](y) = r$.

Thus we define new definition: monotonically probably approximately individually calibrated (mPAIC).

$$P\left(|\bar{h}[X, R](Y) - R| \geq \epsilon\right) \leq \delta \tag{80}$$

Algorithm By following definition of mPAIC, we train the model with below equation.

$$\mathcal{L}_{\text{PAIC}}(\theta) = \frac{1}{n}\sum_{i=1}^{n} \left|\bar{h}_\theta[x_i, r_i](y_i) - r_i\right| \tag{81}$$

Since above equation only gaurantees calibration, we still need to optimize for sharpness using NLL as below.

$$\mathcal{L}_{\text{NLL}}(\theta) = -\frac{1}{n}\sum_{i=1}^{n} \log \frac{d}{dy}\bar{h}_\theta[x_i, r_i](y_i) \tag{82}$$

We balance sharpness and calibration term with weights.

$$\mathcal{L}_\alpha(\theta) = (1-\alpha)\mathcal{L}_{\text{PAIC}}(\theta) + \alpha\mathcal{L}_{\text{NLL}}(\theta) \tag{83}$$

1.7.2 L2 norm

L2 regularization is most common way to regularize the objective by constraining the theta values.

$$\mathcal{L}(\mathbf{w}) = \text{Loss}(\mathbf{X}, \mathbf{y}, \mathbf{w}) + \lambda\|\mathbf{w}\|_2^2 \tag{84}$$

1.7.3 MMD

Maximum Mean Discrepancy (MMD) is a distance on the space of probability measures. It leverages the kernel method to take advantage of localized learning. It also guaranties unbiased estimate since we are using U statistic $h_{i,j}$ D'Andrade [1978]to derive MMD.

$$\mathcal{L}(\mathbf{w}) = \text{Loss}(\mathbf{X}, \mathbf{y}, \mathbf{w}) + \lambda * \text{MMD} \tag{85}$$

$$\text{MMD}(\mathcal{F}, D, Q) = \frac{1}{n(n-1)} \sum_{i=1}^{n} \sum_{j \neq i} h_{ij}, \tag{86}$$

$$h_{ij} \coloneqq k\left((y_i, z_i), (y_j, z_j)\right) + k\left((\hat{y}_i, z_i), (\hat{y}_j, z_j)\right) - k\left((y_i, z_i), (\hat{y}_j, z_j)\right) - k\left((y_j, z_j), (\hat{y}_i, z_i)\right) \tag{87}$$

1.8 Summary of Calibration in regression in terms of distribution matching Marx et al. [2024]

We can formulate calibration in regression method in terms of distribution matching by properly choosing forecast variable and target variable with conditioning variable. We can view as matching between $Y = \hat{Y}$ as $E[Q_{Y|X}(y)] = P_Y(y)$ for all possible y and $Q_{Y|X}(Y) = P_{Y|X}(Y)$ as average distribution matching, which doesn't guarantee calibration in individual instances.

Table 2: Summary of Calibration in Regression

Calibration Objective	**Forecast Variable**	**Target Variable**	**Conditioning Variable**
Quantile Calibration	$Q_{Y\|X}(Y)$	$P_{Y\|X}(Y)$	$-$
Threshold Calibration	$Q_{Y\|X}(Y)$	$P_{Y\|X}(Y)$	$\mathbb{I}\{Q_{Y\|X}(y_0) \leq \alpha\}$
Marginal Calibration	$\hat{Y}$	Y	$-$
Decision Calibration	$\hat{Y}$	Y	$\delta_t^*(Q_{Y\|X})$
Group Calibration	$\hat{Y}$	Y	$\text{Group}(X)$
Distribution Calibration	$\hat{Y}$	Y	$Q_{Y\|X}$
Individual Calibration	$\hat{Y}$	Y	X
Local Calibration	$\hat{Y}$	Y	$\phi(X)$

1.9 Domain Adaptation/Domain Generalization(?)

Problem setup Domain adaptation is a fundamental problem in machine learning that arises when the training and test data come from different distributions, known as the source and target domains, respectively. The goal of domain adaptation is to leverage the knowledge acquired from the source domain to improve the performance of a model on the target domain, where labeled data is often scarce or expensive to obtain.

Formally, let $\mathcal{X}$ be the input space and $\mathcal{Y}$ be the output space. The source domain is defined as $\mathcal{D}_S = \{(\mathbf{x}_i^S, y_i^S)\}_{i=1}^{n_S}$, where $\mathbf{x}_i^S \in \mathcal{X}$ and $y_i^S \in \mathcal{Y}$ are the input and output pairs, respectively, and n_S is the number of labeled samples in the source domain. Similarly, the target domain is defined as $\mathcal{D}_T = \{(\mathbf{x}_j^T, y_j^T)\}_{j=1}^{n_T}$, where $\mathbf{x}_j^T \in \mathcal{X}$ and $y_j^T \in \mathcal{Y}$ are the input and output pairs, respectively, and n_T is the number of samples in the target domain, which may or may not be labeled.

The goal of domain adaptation is to learn a predictive function $f : \mathcal{X} \to \mathcal{Y}$ that minimizes the expected risk on the target domain, $\mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}_T}[\ell(f(\mathbf{x}), y)]$, where ℓ is a loss function, by leveraging the knowledge from the source domain.

Solution approaches Various approaches have been proposed to tackle the domain adaptation problem, For examples, Instance Reweighting **???** aims to assign higher weights to source instances that are more relevant to the target domain, effectively reducing the distribution shift between the domains. Self-Training and Pseudo-Labeling **???** leverage the target data by generating pseudo-labels for the target instances using a model trained on the source domain. The model is then fine-tuned on the combination of source data and pseudo-labeled target data.

Feature Adversarial Representation Learning on the other hand seeks to learn domain-invariant feature representations that bridge the gap between the source and target domains. Inspired by the success of Generative Adversarial Networks (GANs), adversarial domain adaptation methods train a domain discriminator to distinguish between source and target representations, while the feature extractor is trained to confuse the discriminator. Notable works in this category include DANN **?**, ADDA **?**, and CyCADA **?**.

1.9.1 Objective Functions

Comparison of Domain Adaptation Methods: Objective Functions

Notations:

- x_s - source domain data sample
- x_t - target domain data sample
- y_s - source domain label for x_s (if available)
- f - feature extractor
- g - label predictor (classifier)
- h - domain discriminator
- λ - weight hyperparameter

- C - loss function (e.g., cross-entropy)

Domain-Adversarial Training (DANN) [Li et al., 2017]

$$\mathcal{L}_{DANN} = \mathcal{L}_C(f(x_s), y_s) + \lambda \mathcal{L}_{adv}(h, f)$$

- $\mathcal{L}_C$: Classification loss (e.g., cross-entropy) between the source features ($f(x_s)$) and their labels (y_s). Only used if source labels are available.
- $\mathcal{L}_{adv}$: Adversarial loss that encourages the feature extractor (f) to generate domain-agnostic features that fool the domain discriminator (h). It's typically implemented using a gradient reversal layer.

Adversarial Domain Adaptation (ADDA) [Tsai et al., 2019]

$$\mathcal{L}_{ADDA} = \mathcal{L}_C(g(f(x_s)), y_s) + \lambda \mathcal{L}_{adv}(h, f_t)$$

Similar to DANN, but with some key differences:

- The adversarial loss uses a separate feature extractor for the target domain (f_t). This is trained to generate features that are discriminative for the domain discriminator.
- The classification loss is applied to the label predictions after the label predictor (g) instead of directly on the features.

Cycle-Consistent Adversarial Domain Adaptation (CyCADA) [Hoffman et al., 2018]

$$\mathcal{L}_{CyCADA} = \mathcal{L}_C(g(f(x_s)), y_s) + \lambda_1 \mathcal{L}_{adv}(h_s, f_t \circ f_s) + \lambda_2 \mathcal{L}_{adv}(h_t, f_s \circ f_t) + \mathcal{L}_{cyc}(f_s, f_t, x_s, x_t)$$

CyCADA builds upon ADDA by introducing cycle consistency:

- Two domain discriminators (h_s and h_t) are used for each domain.
- Cycle consistency loss ($\mathcal{L}_{cyc}$): Enforces that applying the feature extractors of both domains in sequence (e.g., $f_t \circ f_s$) should bring a sample back to its original domain.

Maximum Classifier Discrepancy (MCD) [Saito et al., 2018]

$$\mathcal{L}_{MCD} = \mathcal{L}_C(g(f(x_s)), y_s) + \lambda \text{MMD}(f(x_s), f(x_t))$$

MCD uses Maximum Mean Discrepancy (MMD) to measure the distance between source and target domain features. Unlike adversarial training, MMD doesn't require a domain discriminator. It directly compares the distributions of source and target features in a kernel space.

In summary:

- Adversarial Training (DANN, ADDA): These methods introduce a domain discriminator. This network tries to distinguish between source and target domain features extracted by the feature extractor. The feature extractor, in turn, tries to "fool" the discriminator by generating features that are indistinguishable between domains. This encourages the feature extractor to learn domain-agnostic representations that capture the underlying task.

- Cycle Consistency (CyCADA): CyCADA builds upon the adversarial approach of ADDA, but additionally enforces *cycle consistency*. It utilizes two domain discriminators and forces the feature extractors to be inverses of each other. This means applying feature extractors of both domains sequentially should bring a sample back to its original domain. This further strengthens the domain-agnostic nature of the features.
- Maximum Mean Discrepancy (MCD): MCD takes a different approach. It doesn't rely on adversarial training. It directly compares the distributions of source and target domain features in a kernel space using *Maximum Mean Discrepancy (MMD)*. This distance measure penalizes the model when the feature distributions diverge significantly. By minimizing MMD, the model learns representations that are closer between domains, promoting alignment.

1.9.2 Datasets Used in Experiments

- MNIST **?**: A handwritten digit dataset, commonly used for domain adaptation between different digit datasets.
- SVHN **?**: The Street View House Numbers dataset, consisting of colored digit images from Google Street View.
- USPS **?**: The United States Postal Service dataset of handwritten digits.
- Office-31 **?**: A benchmark dataset for visual domain adaptation, containing 4,652 images from 31 object categories in three distinct domains: Amazon (product images), Webcam, and DSLR.
- Office-Home **?**: An extension of Office-31, with 15,588 images from 65 categories across four domains: Artistic, Clipart, Product, and Real-World.
- VisDA-2017 **?**: A large-scale dataset for visual domain adaptation, containing over 280,000 images across two domains: synthetic renderings and real images.

These datasets are widely used in the domain adaptation literature to evaluate the performance of different methods across various tasks, such as digit recognition, object recognition, and image classification.

1.10 Fairness

1.10.1 Types of Bias (Mehrabi et al. [2021])

1. Measurement Bias
2. Omitted Variable Bias
3. Representation Bias
4. Aggregation Bias
5. Sampling Bias
6. Longitudinal Data Fallacy
7. Linking Bias

1.11 Definition of Fairness

•

1.12 Questions

- What are the most stable methods
  - Hyperparameter tuning
- Is there a strong correlation with distribution matching in their specific tasks?
- Uniqueness of mapping
  - Are you learning a map that just shift or is it a unique mapping?

1.13 My Questions

- How would you formulate the general framework of distribution matching that aligns with all works?
- what are the most robust models?
- What are the most stable metrics?
- What area has actually seen progress aligned with progress of research
- Find methodologies corresponding to each task that compares distrubition matching with trustworthy ML.

Bibliography

Alexandru Niculescu-Mizil and Rich Caruana. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd international conference on Machine learning*, pages 625–632, 2005.

Deng-Bao Wang, Lei Feng, and Min-Ling Zhang. Rethinking calibration of deep neural networks: Do not be afraid of overconfidence. *Advances in Neural Information Processing Systems*, 34: 11809–11820, 2021.

Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 29, 2015.

Ananya Kumar, Percy S Liang, and Tengyu Ma. Verified uncertainty calibration. *Advances in Neural Information Processing Systems*, 32, 2019.

Jarosław Błasiok, Parikshit Gopalan, Lunjia Hu, and Preetum Nakkiran. A unifying theory of distance from calibration. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pages 1727–1740, 2023.

Jarosław Błasiok and Preetum Nakkiran. Smooth ece: Principled reliability diagrams via kernel smoothing. *arXiv preprint arXiv:2309.12236*, 2023.

Oded Goldreich, Shari Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *Journal of the ACM (JACM)*, 45(4):653–750, 1998.

Jaroslaw Blasiok, Parikshit Gopalan, Lunjia Hu, and Preetum Nakkiran. When does optimizing a proper loss yield calibration? *Advances in Neural Information Processing Systems*, 36, 2024.

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

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

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

Volodymyr Kuleshov, Nathan Fenner, and Stefano Ermon. Accurate uncertainties for deep learning using calibrated regression. In *International conference on machine learning*, pages 2796–2804. PMLR, 2018.

Roshni Sahoo, Shengjia Zhao, Alyssa Chen, and Stefano Ermon. Reliable decisions with threshold calibration. *Advances in Neural Information Processing Systems*, 34:1831–1844, 2021.

Hao Song, Tom Diethe, Meelis Kull, and Peter Flach. Distribution calibration for regression. In *International Conference on Machine Learning*, pages 5897–5906. PMLR, 2019.

Tilmann Gneiting and Matthias Katzfuss. Probabilistic forecasting. *Annual Review of Statistics and Its Application*, 1:125–151, 2014.

Geoff Pleiss, Manish Raghavan, Felix Wu, Jon Kleinberg, and Kilian Q Weinberger. On fairness and calibration. *Advances in neural information processing systems*, 30, 2017.

Moritz Hardt, Eric Price, and Nati Srebro. Equality of opportunity in supervised learning. *Advances in neural information processing systems*, 29, 2016.

Jon Kleinberg, Sendhil Mullainathan, and Manish Raghavan. Inherent trade-offs in the fair determination of risk scores. *arXiv preprint arXiv:1609.05807*, 2016.

Shengjia Zhao, Tengyu Ma, and Stefano Ermon. Individual calibration with randomized forecasting. In *International Conference on Machine Learning*, pages 11387–11397. PMLR, 2020.

Roy G D'Andrade. U-statistic hierarchical clustering. *Psychometrika*, 43:59–67, 1978.

Charlie Marx, Sofian Zalouk, and Stefano Ermon. Calibration by distribution matching: trainable kernel calibration metrics. *Advances in Neural Information Processing Systems*, 36, 2024.

Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning. *ACM computing surveys (CSUR)*, 54(6):1–35, 2021.