Probability Calibration

David S. Rosenberg

NYU: CDS

April 14, 2021

Contents

Probability calibration: informal

What do we mean by calibration?

- Our predictions are calibrated if
  - we look at all the events predicted to occur with probability $X\%$, and
  - about $X\%$ of these events actually occur.

When don't we care about calibration?

- We just want the best guess (most likely) classification for any input.
    - (hard classification will suffice)
- From a large group, we want to find the most likely to be in a particular class.
    - a ranking or any numeric score will suffice (e.g. from an SVM)

When should we care about calibration?

- Decision may depend on the probability.
- For example,
  - should we give a loan? (depends on probability of default)
  - should we insure you? (depends on probability of incident)
- Assess model impact via simulation
  - e.g. if we make a sales call to person A, they are 10% likely to buy the product.
- Fairness / bias
  - Is model more/less confident for some cases than for others?

What does it mean to "calibrate" a model?

- Calibrating a model generally means creating a new model that
  - takes the model prediction as input (anything numeric) and
  - produces a calibrated probability as output
- On the one hand,
  - this is as simple as a probabilistic binary classification problem with a single input feature.
- BUT, depending on the criteria we use for **assessing** the calibration of our "calibrated" model,
  - we might need serious restrictions on the model that we use.

Assessing calibration

Approach 1: Proper scoring rules

- Suppose we're trying to predict the probability that $Y = 1$.
- Suppose we have a loss that's a proper scoring rule.
- The Bayes optimal prediction function w.r.t. this loss
    - will be the true conditional probability $x \mapsto \mathbb{P}(Y = 1 \mid X = x)$.
- Although we don't yet have a formal definition yet, by any reasonable definition,
    - $x \mapsto \mathbb{P}(Y = 1 \mid X = x)$ should be considered "perfectly calibrated."
- This motivates using **hold-out loss** for some proper scoring rule
    - as a measure of calibration.
- Two most common: log-loss and square loss (i.e. Brier score)

Recall that the Bayes optimal prediction function is the function that minimizes the risk, i.e. the expected loss, over all functions of the input space.

"Log-loss" (i.e. negative log-likelihood)

- Suppose we predict an event will happen with probability p.
  - If it happens, the log loss is $-\log p$.
  - If it doesn't happen, the log loss is $-\log(1-p)$.
- The log-loss on the test set is just the mean of the log-loss on each item.
- Suppose we're predicting a rare event, and it happens.
  - If we predict $p = 0.01$, log loss is $-\log(0.01)$
  - If we predict $p = 0.001$, the log loss is $-\log(0.001)$.
  - The difference is $-\log(0.001) - [-\log(.01)] = \log\left(\frac{.01}{.001}\right) = \log(10)$.
- The difference between the log-losses depends on the ratio of the probabilities.
- Note also that $-\log(0) = \infty$... stay safe by predicting $p \in [\varepsilon, 1-\varepsilon]$ for some $\varepsilon > 0$.

- Certain types of models, such as logistic regression and neural networks, can end up predicting extremely small or large probabilities when extrapolating beyond the support of the training data. You can use common sense in setting a lowest possible and largest possible predicted probability. For example, if you have only 1000 training example, you can't possibly have support for predicting a probability less than 1/1000, and even that is probably much too small.
- This is especially important in predicting rare events, in which case it's usually safer to predict too large a probability than too small a probabililty.

Brier score / square loss

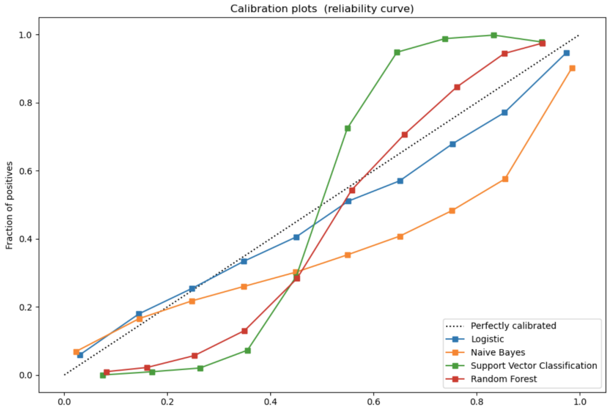
- Suppose we predict a rare event and it happens:
    - If we predict $p = 0.01$, square loss is $(1 - .01)^2 = .980$
    - If we predict $p = 0.001$, square loss is $(1 - .001)^2 = 0.998$
- Both are near the worst-case loss of 1, but essentially the same.

Square loss vs log-loss

- If we're talking about rare events, e.g.
  - probability of defect, loan default, engine failure...
- The difference between $p = .01$ and $p = .001$ can be HUGE.
  - Can change the expected cost by 10x.
- Use log-loss AND make sure to lower-bound the probabilities with something reasonable.
- If we're predicting the total expected number of events, e.g.
  - e.g. $\mathbb{E}\left[\text{number of voters}\right] = \sum_{i=1}^{n} \mathbb{P}\left(\text{voter } i \text{ will vote}\right)$
  - The difference between $p_i = .01$ and $p_i = .001$ is probably negligable.
  - While the difference between $p = 0.2$ and $p = 0.4$ quite important.
- In these situations, use square loss.

Approach 2: Visually with binning

sci-kit learn provides "calibration plots" / "reliability diagrams"



- We've taken a set of examples and partitioned them into bins (partitions are typically either equally spaced or done at quantiles).
- Each dot corresponds to a bin
  - The x-coordinate is the mean predicted probability for elements of the bin (one can also use other appracohes to get a probability prediction for the bin as a whole)
  - The y-coordinate is the fraction of elements in the bin for which the event actually occurs
- Perfect calibration would be the line $y = x$.

Approach 3: Estimate direct measure of calibration

- Later we'll give a formal definition of "calibration error."
- For now, it's basically the average "distance" (e.g. square or absolute distance)
    - between the calibration curve and $y = x$.

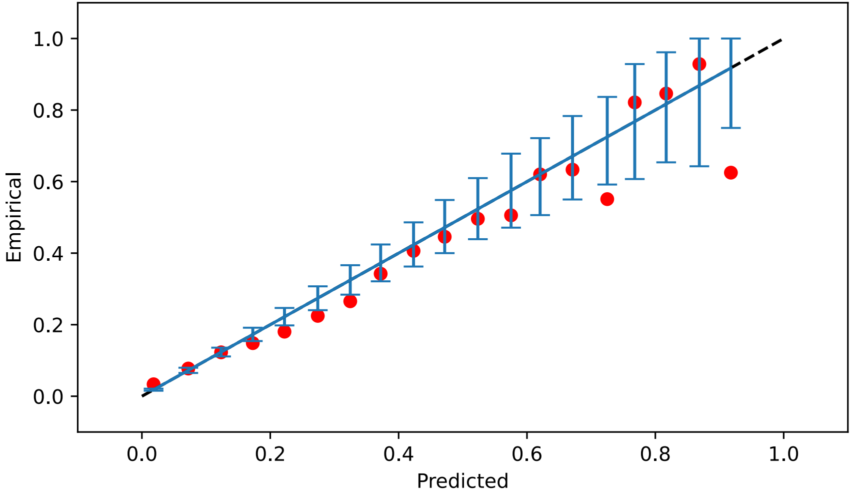
Example: Random forest fit[1] (I)

- Split data into 3 chunks:
  - train, calibration, test
- We fit a random forest on the training set.
- On the test set, the AUROC is 0.765.
- Not bad.
- But this tells us nothing about whether the predicted probabilities are calibrated.
- AUROC depends only on the order of the items in the test set determine of the predicted scores

[1]Based on Brian Lucena's Calibration Workshop Jupyter notebook.

Baseline performance on test

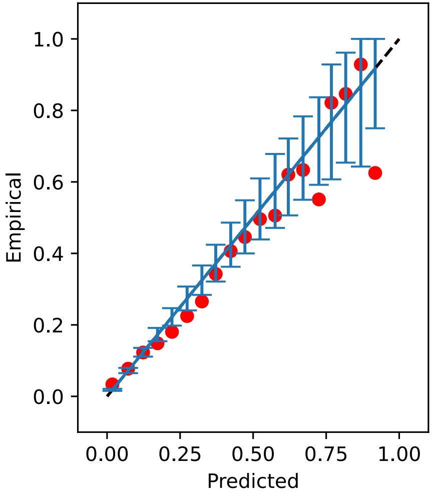
- AUROC: 0.761 Log-loss: 0.388 Square loss: 0.085

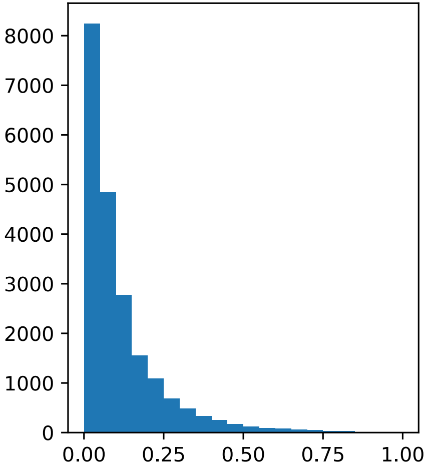


- Predicting 0.7 for a case that occurs gives a square loss of $(1-0.7) = .09$. This gives some perspective on what a mean square loss of 0.085 means.
- This figure is from Calibration Workshop Jupyter notebook and uses the plot_reliability_diagram in the ml_insights package.
- For each bin, we have an average predicted probability p. That is described by the x-position of the red dots and blue intervals. The y-value of the red dot indicates the actual proportion of events occurring in the bin. For each bin, we want to test the null hypothesis that the bin is calibrated – that is, that the actual probability for events in the bin is p, as predicted. The acceptance region for that hypothesis is indicated by the blue interval – it depends on how many points are in the bin and p. If the red dot lies outside the blue interval for any bin, we reject the hypothesis that the bin is calibrated.
- As a not-super-important technical point, the blue intervals are **not** confidence intervals. The interval is not based on data – it's based on a prediction p and the amount of data we have in that interval. We could put a confidence interval on the red dot – these would be confidence intervals for the probability of the event for the bin, in which case we'd want to see if that interval contains p or, equivalently, intersects the line $y = x$.

Reliability diagram with histogram

- AUROC: 0.761 Log-loss: 0.388 Square loss: 0.085

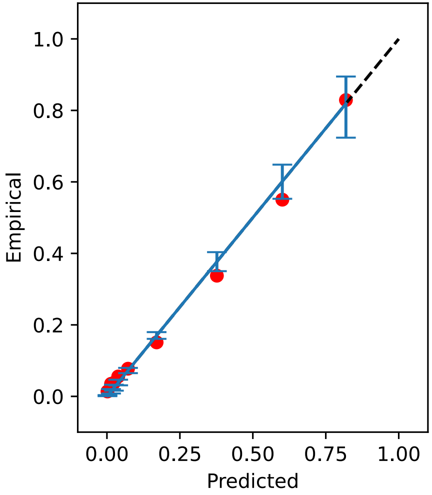


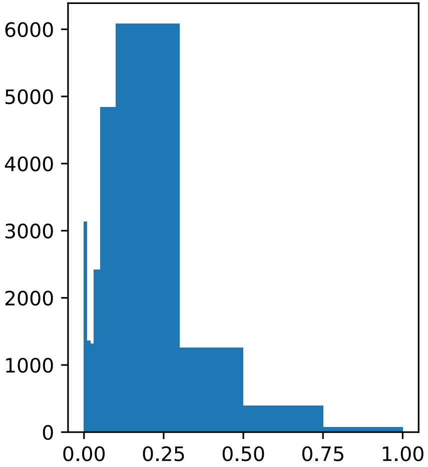


- This figure is from Calibration Workshop Jupyter notebook and uses the plot_reliability_diagram in the ml_insights package (show_histogram=True)
- We now include a graph showing the number of observations in each bin. You can see that the length of the acceptance intervals get wider as we have less data.
- In practice, one usually wants to see both of these graphs to interpret the results. If we have great calibration in regions that are very rare and poor calibration in common areas, that's a serious issue. While poor calibration in regions that rarely occur may be irrelevant.
- These are equally spaced bins – for this data, it probably makes more sense to merge the bins without much data...

Reliability diagram with new bins

- AUROC: 0.761 Log-loss: 0.388 Square loss: 0.085

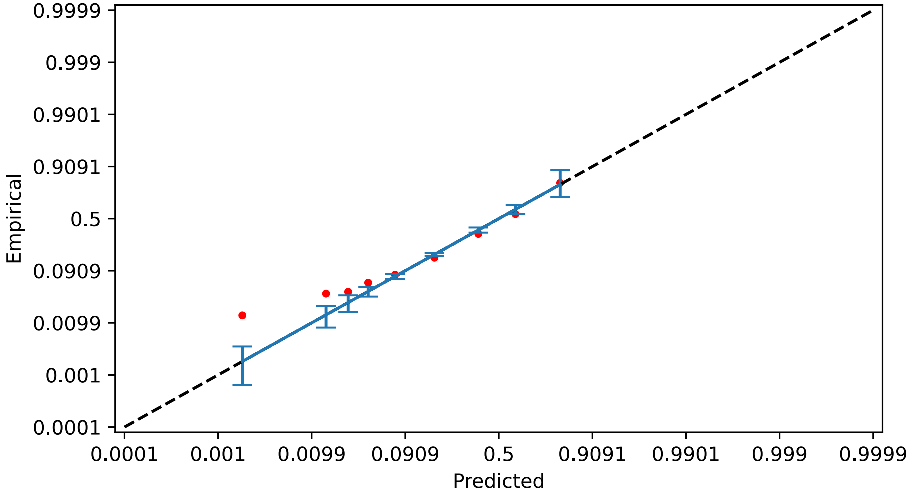




- This figure is from Calibration Workshop Jupyter notebook and uses the plot_reliability_diagram in the ml_insights package (show_histogram=True)
- We now include a graph showing the number of observations in each bin. You can see that the length of the acceptance intervals get wider as we have less data.
- In practice, one usually wants to see both of these graphs to interpret the results. If we have great calibration in regions that are very rare and poor calibration in common areas, that's a serious issue. While poor calibration in regions that rarely occur may be irrelevant.
- These are equally spaced bins – for this data, it probably makes more sense to merge the bins without much data...

Reliability diagram with logit scaling

- AUROC: 0.761 Log-loss: 0.388 Square loss: 0.085

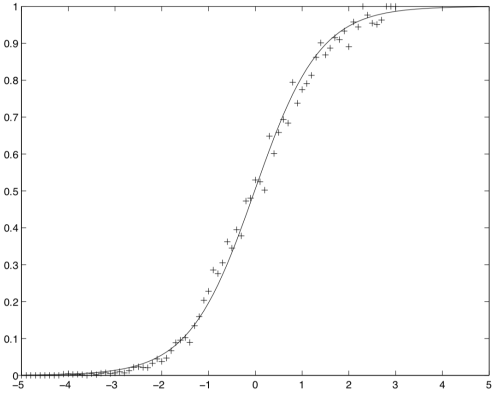


- Now we can see what's going on in the low probability regions better. Calibration doesn't look great down there. The probabilities are consistently lower than they should be for small probabilities, and higher than they should be for probabilities in the middle range.

Calibration methods

Platt scaling (sklearn and most people's version)

- Fit[2] a logistic regression on $(f(X_i), Y_i)_{i=1}^N$.



[2]Figure 2 from [Pla99]

- The logisitic regression is fit to data of the form $(f(x), y)$, where $f(x) \in \mathbb{R}$ is the prediction of the original classifier, and $y \in \{0, 1\}$ is the true class label.
- What's plotted here is one point for each bin – the bins are of width 0.1. The x and y axes are exactly as in the previous reliability diagrams.
- Although this is what everybody calls Platt scaling, if you actually read the paper, you'll find they suggest modifying the 0/1 labels to be soft labels, to reduce overfitting. sklearn can't fit this kind of model, but mathematically it doesn't require any change from fitting logistic regression. You can do it in PyTorch easily. See [Pla99] for details.

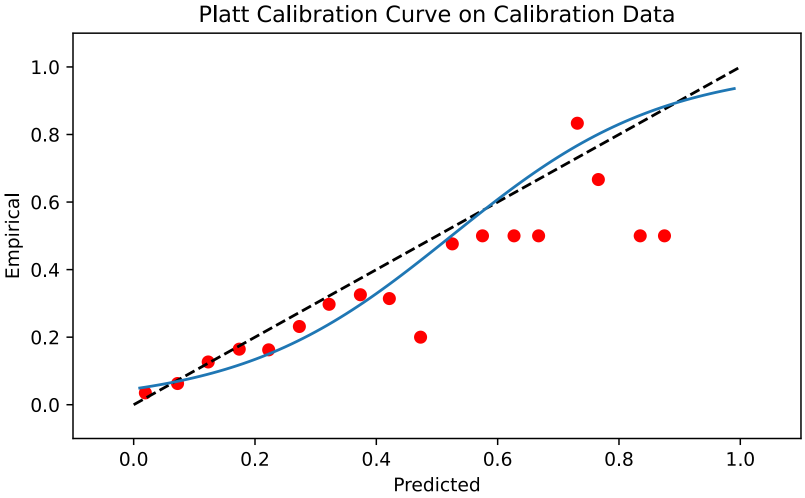
Platt scaling

- Plat scaling assumes a very specific form:

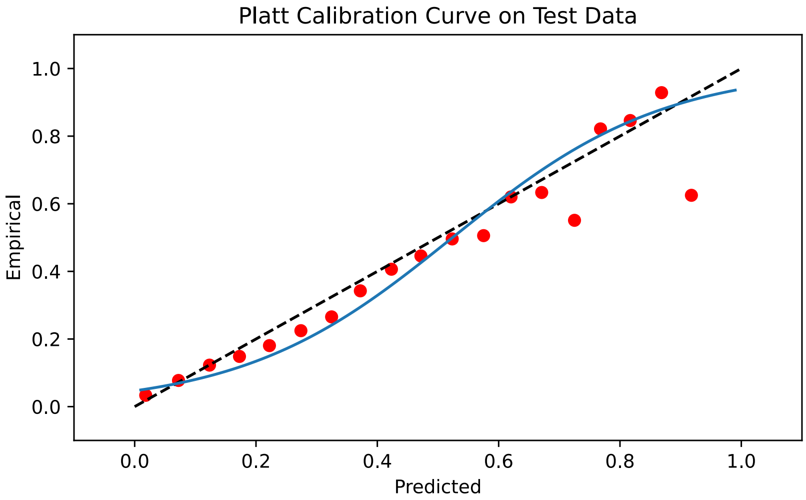
$$\mathbb{P}(Y = 1 \mid f(x)) = \frac{1}{1 + \exp(af(x) + b)}.$$

- Advantage: Only 2 parameters – needs relatively little data to fit.
- Disadvantage: Basically assumes $f(x)$ is on the logit scale.
  - If $f(x)$ is a perfectly calibrated probability, Platt scaling will mess it up.
  - Unless you're smart and first transform $f(x)$ to the logit scale: $f(x) \mapsto \log\left(\frac{f(x)}{1-f(x)}\right)$.
- Platt scaling gives a smooth and strictly increasing calibration
  - so rankings are preserved (i.e. has no affect on AUROC)

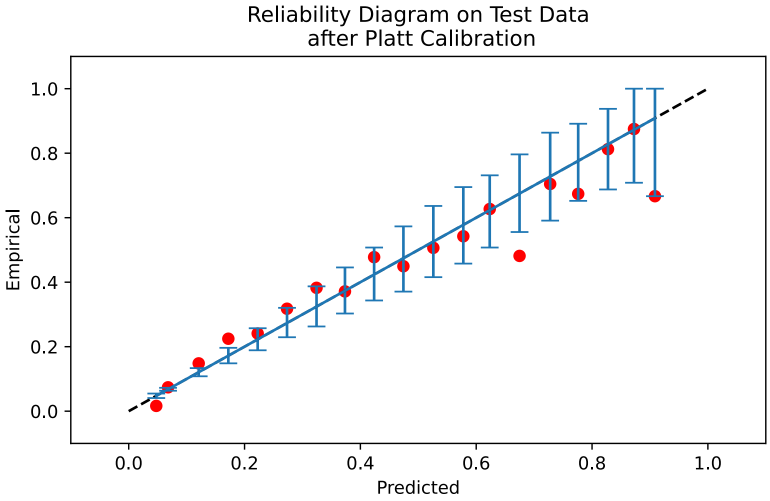
Platt scaling: on calibration data



Platt scaling: on test data



Platt scaling: reliability diagram on test data



Platt scaling: results summary

- Uncalibrated Brier score: 0.0849
- Platt calibrated Brier score: 0.0851
- Uncalibrated log loss: 0.313
- Platt calibrated log loss: .298

Other methods

- Isotonic regression: piecewise constant; non-decreasing; needs lots of data
- Histogram method: piecewise constant; needs lots of data
- Spline calibration: smooth, nonparameteric [Luc18]
- Generalized additive models: smooth, nonparametric
- Beta calibration: parametric, smooth, somewhat less restrictive than Platt

See Brian Lucena's workshop for a patient walk-thru of many of these.

What's special about calibration as an ML problem?

- Should use a proper scoring rule for a loss function (e.g. log-loss or square loss)
- Has just one input feature
- Sometimes you don't have much data to fit it
- We expect the calibration curve to roughly non-decreasing
    - though not necessarily – often extreme values have weird behavior
- So... some stuff... but most ML methods should be reasonable to try for calibration.

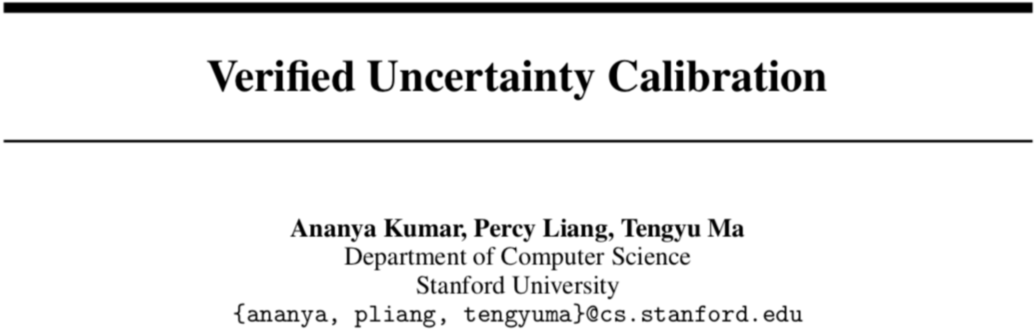
The standard multiclass hack

- There's a standard hack for calibrating multiclass probability predictions.
- Calibrate probabilities for each class separately, using one-vs-all.
- Then renormalize the predictions into a probability distribution.
- How to assess?
- Ver 1: Only care about the calibration of the most likely class
- Ver 2: Combine calibration scores across classes (e.g. by averaging)

Formalization of calibration

Main reference for rest of talk

- The rest of this talk is mostly based on [KLM19]



Verified Uncertainty Calibration

Ananya Kumar, Percy Liang, Tengyu Ma
Department of Computer Science
Stanford University
`{ananya, pliang, tengyuma}@cs.stanford.edu`

What is calibration?

- Informally, events we predict with probability $X\%$ should occur $X\%$ of the time.

Definition (Perfect calibration)

We say a forecaster $f : \mathcal{X} \to [0,1]$ is perfectly calibrated if for all $p \in [0,1]$:

$$\mathbb{P}[Y = 1 \mid f(X) = p] = p.$$

- So are we thrilled with any forecaster that is perfectly calibrated?

Do we like any forecaster that is perfectly calibrated?

Example (Best constant forecaster)

Consider the predictor $f(x) \equiv \mathbb{P}(Y = 1)$. It only outputs a single probability, regardless of x. Also note that it's perfectly calibrated.

- Calibration is appealing, but it isn't enough to make a satisfactory forecaster.
- We want something reasonably well-calibrated **and** with good score performance.

Something too strong...

- The absolute best we can do is predict $\mathbb{P}[Y = 1 \mid X]$.
- We could consider the **integrated squared error**

$$\operatorname{ISE}(f) = \left(\mathbb{E}\left[(f(X) - \mathbb{P}[Y = 1 \mid X])^2\right]\right)^{1/2}.$$

- But $x \mapsto \mathbb{P}[Y = 1 \mid X = x]$ can be an extremely complicated function of x.
- We may have no chance to learn it.
- As we discuss below, we have no estimate ISE without assumptions.

Calibration error (CE)

Definition (Calibration error)

The **calibration error** of $f : \mathcal{X} \to [0,1]$ is given by

$$\mathrm{CE}(f) = \left(\mathbb{E}\left[\left(f(X) - \mathbb{P}[Y = 1 \mid f(X)]\right)^2 \right] \right)^{1/2}$$

- A well-calibrated $f(x)$ will have $\mathbb{P}[Y = 1 \mid f(X)] \approx f(X)$
- CE is just the expected square difference between them (square-rooted)
- Sanity check: does perfect calibration correspond to 0 calibration error?
- Note: This is a a property of the distribution – we still have to estimate $\mathrm{CE}(f)$ from data...

What do really want?

- We can get great calibration by ignoring x and predicting $\mathbb{E}[Y]$.
- But the whole point is to make more accurate probability predictions using x.
- With infinite data and any proper score function, we'd end up with $f(x) = \mathbb{E}[Y \mid X = x]$.
- But we don't have infinite data.
- In practice, we start with an f that optimizes a scoring rule...
- Then we mess with it to try to get the calibration better, without messing up the score performance too much.

Cool decomposition of the Brier score / mean squared error

- For convenience, define $t(p) := \mathbb{E}[Y \mid f(X) = p] = \mathbb{P}[Y = 1 \mid f(X) = p]$.
- We say "t" for "target": for perfect calibration, $f(x) \equiv t(x)$.
- With some math, we can show:

$$\mathbb{E}\left[(Y - f(X))^2\right] = \underbrace{\mathrm{Var}(Y)}_{\text{uncertainty}} - \underbrace{\mathrm{Var}(t(X))}_{\text{sharpness}} + \underbrace{\mathbb{E}\left[(t(X) - f(X))^2\right]}_{\text{calibration error}}.$$

Evaluating / Verifying Calibration

We can define CE, but can we evaluate it?

Claim: We cannot evaluate CE for a general $f(x)$

- The hard part is estimating $\mathbb{P}(Y = 1 \mid f(X) = \hat{p})$ for all possible $\hat{p}$.
- For each $\hat{p}$, need a sample of (X, Y)'s that have $f(X) = \hat{p}$.
- Then we can estimate $t(x)$ using the empirical fraction with $Y = 1$.
- But we need such a sample for every distinct value of $f(X)$.
- If there are infinitely many distinct value of $f(X)$ we're out of luck.
  - Unless we make some additional assumptions...

What assumptions allow estimating CE?

- You basically have to assume that examining just a finite number of $f(X)$'s is enough.
- e.g. Assuming some smoothness of $\mathbb{P}(Y = 1 \mid f(X) = \hat{p})$ as a function of $\hat{p}$.
- OR
- Making sure there are only a finite number of $f(X)$'s.
- I only know of people taking the second approach.

The main approach: binning

Definition (Binning)

A binning $\mathcal{B}$ is a partition of $[0,1]$ into disjoint sets $I_1, \ldots, I_B$.

- Typically the I_b's are intervals.

Definition (Binned version of f)

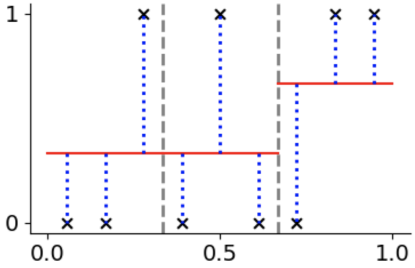
The binned version of $f : \mathcal{X} \to [0,1]$ for binning $\mathcal{B}$ is

$$f_{\mathcal{B}}(x) = \mathbb{E}[f(X) \mid f(X) \in I_j] \qquad \text{where } x \in I_j$$

- In practice, we estimate this with a sample $X_1, \ldots, X_n$ by defining $\hat{f}_{\mathcal{B}}(x)$ to be the average of all $f(X_i)$ where X_i and x are in the same interval.

Binning example

- Below we've partitioned $[0,1]$ into 3 intervals
- The red lines show the value of the binned $f_{\mathcal{B}}$ on each interval
- If the x's are held out points, how's our calibration[3]?



[3]Figure 1(b) from [KLM19]

Basic approach to estimating calibration of general f

- Choose a binning, usually by equally sized quantiles
    - so each bin as same amount of data
- Compute average $f(X_i)$ for data in each bin
    - This estimates $f_{\mathcal{B}}$ for each bin
- Compute average Y_i for data in each bin
    - This is exactly estimating a Bernoulli rate from coin flips
    - This estimates $\mathbb{E}[Y \mid f(X)]$
- The difference in averages is the estimate CE for each bin
- Overall CE is the RMS average of these bin CE's

Formal definition of the plugin estimator of CE

- The **plug-in** estimator of CE is $\hat{\mathcal{E}}^2_{\text{pl}} = \sum_{s \in S} \hat{p}_s (s - \hat{y}_s)^2$ where
    - let each bin be identified by s, the average value of $f(x)$ in the bin
    - $\hat{p}_s$ is the relative size of the bin (i.e. the bin's weight)
    - s is the probability prediction for the bin
    - $\hat{y}_s$ is the estimate of the probability of outcome in bin

Bias/variance tradeoff in estimating CE

- The SE for the estimate of the Bernoulli rate in any bin is $\sqrt{(}p(1-p)/n)$.
- Fewer bins $\implies$ larger n in each bin $\implies$ better estimates of $\mathbb{E}[Y \mid f_{\mathcal{B}}(X)]$
- Fewer bins $\implies$ $f_{\mathcal{B}}(x)$ is a worse approximation of $f(x)$.
- This is a fairly classic bias/variance tradeoff.

Theorem (Binning underestimates error)

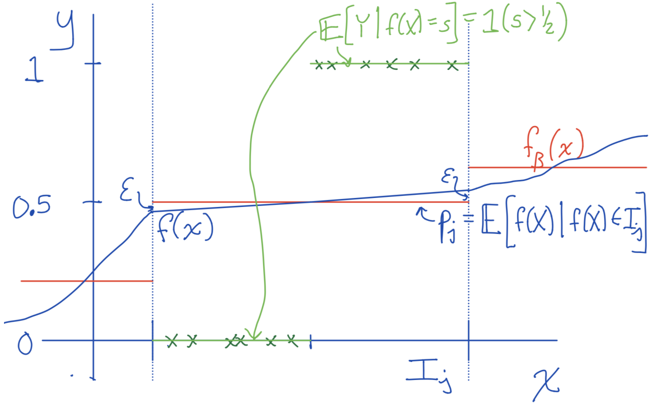
For any binning scheme $\mathcal{B}$ and model $f : \mathcal{X} \to [0,1]$ we have

$$CE(f_{\mathcal{B}}) \leqslant CE(f).$$

- Note: There is no statistical estimation in this theorem. Just approximation of f by $f_{\mathcal{B}}$.

Demonstration of underestimation

- $f(x)$ is very badly calibrated on I_j: very close to 0.5, but actual prob is 0/1
- $f_{\mathcal{B}}(x)$ perfectly calibrated on I_j.



y
1
0.5
0
ε_1
ε_2
$f(x)$
$f_B(x)$
$\mathbb{E}[Y \mid f(X)=s] = 1(s > \frac{1}{2})$
$p_j = \mathbb{E}[f(X) \mid f(X) \in I_j]$
I_j
x

Two ways to think about binning

Binning is just a way to estimate CE of f

We like our smooth $f(x)$. But of course we have to bin it to estimate its CE. We get our estimate of $f_{\mathcal{B}}(x)$, but we deploy $f(x)$ and claim its CE is not much worse than CE of $f_{\mathcal{B}}(x)$.

Binned f is the only thing that matters

Our original $f(x)$ is irrelevant: its CE can be arbitrarily bad, no matter how good the CE of a binned version. We deploy a binned version $f_{\mathcal{B}}(x)$ that achieves a reasonable CE. We forget about $f(x)$ for anything requiring calibration.

Unbiased estimator of CE

- Turns out the plug-in estimator is biased.
- The debiased estimator for squared calibration error is

$$\hat{\mathcal{E}}^2_{\text{db}} = \sum_{s \in S} \hat{p}_s \left[(s - \hat{y}_s)^2 - \frac{\hat{y}_s(1 - \hat{y}_s)}{\hat{p}_s n - 1} \right]$$

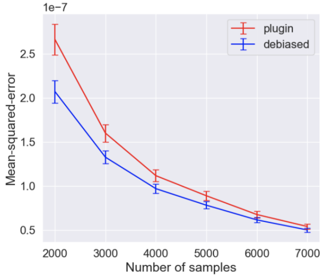
- Performance is better.

Experiments on biased vs debiased CE estimators

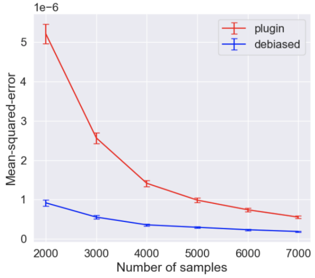
- On CIFAR-10, validation set is size 10000
- Split validation set into S_C and S_E of sizes 3000 and 7000, respectively.
- Use S_C to re-calibrate and discretize a trained VGG-16 model.
- Use S_E to estimate calibration error
- Each of the $K = 10$ classes is calibrated separately.
- We calibrate with either $B = 100$ or $B = 10$ bins per class.
- Take CE on S_E set (size 7000) as ground truth
- Compare biased and debiased estimators to "ground truth" for various subsample sizes.

Experiments on biased vs debiased CE estimators: results

From [KLM19] (confidence intervals from bootstrapping)



(a) $B = 10$



(b) $B = 100$

Figure 4: Mean-squared errors of plugin and debiased estimators on a recalibrated VGG16 model on CIFAR-10 with 90% confidence intervals (lower values better). The debiased estimator is closer to the ground truth, which corresponds to 0 on the vertical axis, especially when B is large or n is small. Note that this is the MSE of the squared calibration error, not the MSE of the model in Figure 3.

With fewer bins on the left, the MSE is about an order of magnitude lower. This makes sense: we have an order of magnitude more data per bin, and variance (basically MSE without the bias) decreases linearly with sample size.

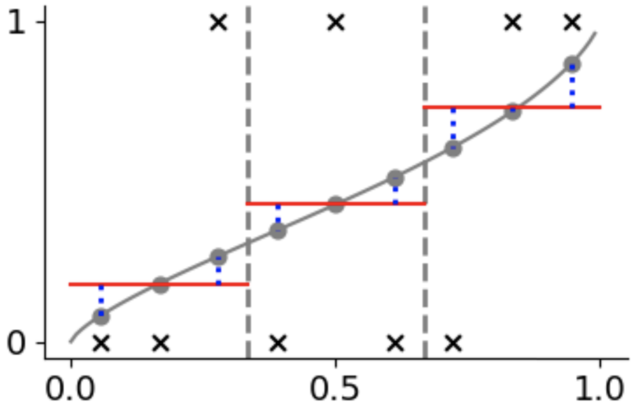
More calibration techniques

Histogram binning

- The same process as estimating calibration error by binning
- The "recalibrated" probability for a bin is an estimate of the probability of $Y = 1$ in that bin
- What data do we use for this calibration:
  - Theory: a hold out calibration set
  - Practice 1: Calibrate using out-of-sample predictions from CV in training set
  - Practice 2: Just re-use training set (only works if we don't have overfitting on training set) [NCH14]
- Histogram binning requires a lot of data
  - scales linearly with the number of bins
- Also: "often want to recalibrate our models in the presence of domain shift or recalibrate a model trained on simulated data, and may have access to only a small labeled dataset from the target domain." [KLM19]

Scaling-binning calibrator: picture

By picture[4]:



[4]Image from Fig 1(c) of [KLM19]

Scaling-binning calibrator: idea

- Start with some “scaling” calibrator, such as Platt scaling
    - This is really just any calibrator that’s not discretized enough for direct evaluation of CE
- Then create a binned version of that
- The Platt scaling calibration has values that are much closer together than the raw 0/1 observations
- Leads to much lower variance estimates of the calibrated output
- Of course, it may be quite biased by the Platt scaling piece
- Authors would argue, you can’t really say anything about Platt scaling’s CE anyway

References

Resources

- Brian Lucena's "Probability Calibration Workshop" videos.

References I

[KLM19] A. Kumar, P. Liang, and T. Ma, *Verified uncertainty calibration*, Advances in Neural Information Processing Systems (NeurIPS), 2019.

[Luc18] Brian Lucena, *Spline-based probability calibration*, CoRR (2018).

[NCH14] Mahdi Pakdaman Naeini, Gregory F. Cooper, and Milos Hauskrecht, *Binary classifier calibration: Non-parametric approach*, CoRR (2014).

[Pla99] John C. Platt, *Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods*, ADVANCES IN LARGE MARGIN CLASSIFIERS, MIT Press, 1999, pp. 61–74.