Probability Calibration

David S. Rosenberg

NYU: CDS

November 17, 2021

Contents

1. Probability calibration: informal
2. Assessing calibration
3. Calibration methods
4. Formalization of calibration error
5. Estimating calibration error

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.
  - i.e. 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)
- e.g. Expected return for customer x
  - Let $f(x)$ be the amount that x will spend if they buy a product.
  - Let $p(x)$ be the probability that x will buy a product if make a sales call
  - If $\hat{p}(x)$ is a calibrated estimate of $p(x)$, then
    - expected return for a sales call to x is roughly $\hat{p}(x)f(x)$.
  - We should call customers with highest $\hat{p}(x)f(x)$.
- 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 (some scalar) as input and
  - produces a calibrated probability as output.
- This is a probabilistic binary classification problem with a single input feature.
- **Assessing** the calibration of our "calibrated" model may be a newer topic for you.
- Certain assessment methods may make strong restrictions on the model that we use.
  - We'll elaborate on this mysterious statement in a later section.

- The basic issue being referred to in the last bullet is that, for a certain definition of "calibration error", we can only give unbiased performance estimates when there are only finite set of distinct outputs ever produced by the calibrated model.

Generic approach to building a calibrated model

- Split data into 3 sets: train, calibration, test
- Build our model on the training set.
- Fit our calibration model using the calibration set.
- Assess the calibration using the test set.
- Instead of a separate calibration set, you can also do one of the following:
  1. Calibrate using out-of-sample predictions from CV on training set [Pla99, p. 6]
  2. Re-use the training set (only works if we don't have overfitting on training set) [NCH14]

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 is
  - the true conditional probability $x \mapsto \mathbb{P}(Y = 1 \mid X = x)$.
- Although we don't yet have a formal definition, by any reasonable definition,
  - $x \mapsto \mathbb{P}(Y = 1 \mid X = x)$ should be considered "perfectly calibrated."
- This motivates using the **hold-out loss** of 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 the probability of a **rare event**, and it happens.
  - If we predicted $p = 0.01$, log loss is $-\log(0.01)$
  - If we predicted $p = 0.001$, 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 not conservative enough.
- This is especially important in predicting rare events (e.g. identifying a network security breach of credit card fraud), in which case we would usually prefer to predict too large a probability (leading to a false positive) than too small a probability (leading to false negative).

Brier score / square loss

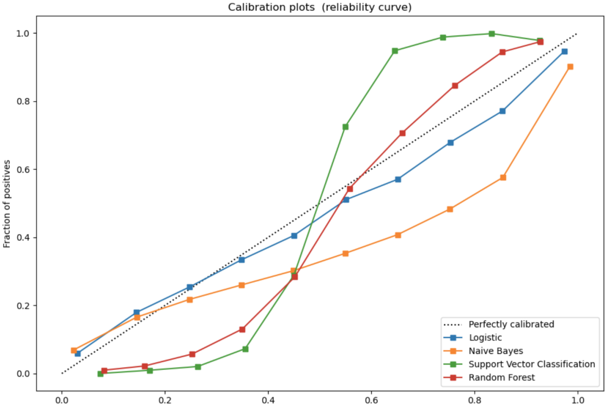
- Suppose we're predicting the probability of a rare event, and it happens.
  - If we predict $p = 0.01$, square loss is $(1 - .01)^2 = .980$
  - If we predict $p = 0.001$, square loss is $(1 - .001)^2 = 0.998$
- Both are near the worst-case loss of 1, but essentially the same.

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 expected cost by $10\times$.
- 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 $p = 0.2$ and $p = 0.4$ is much more important than
  - difference $p_i = .01$ and $p_i = .001$.
- 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 approaches 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 $y = x$.

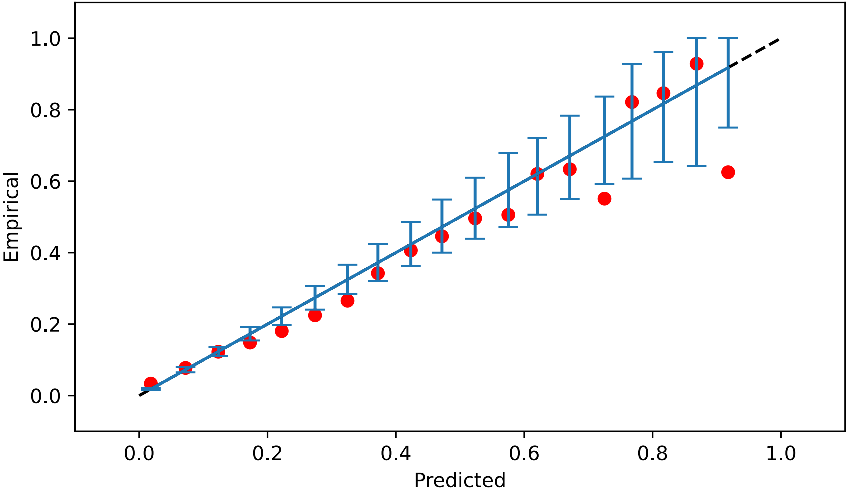
Example: Random forest fit[1] (I)

- 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 ranking by score of the items in the test set

[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 $(1-0.7)^2 = .09$. This gives some perspective on what a mean square loss of 0.085 means.

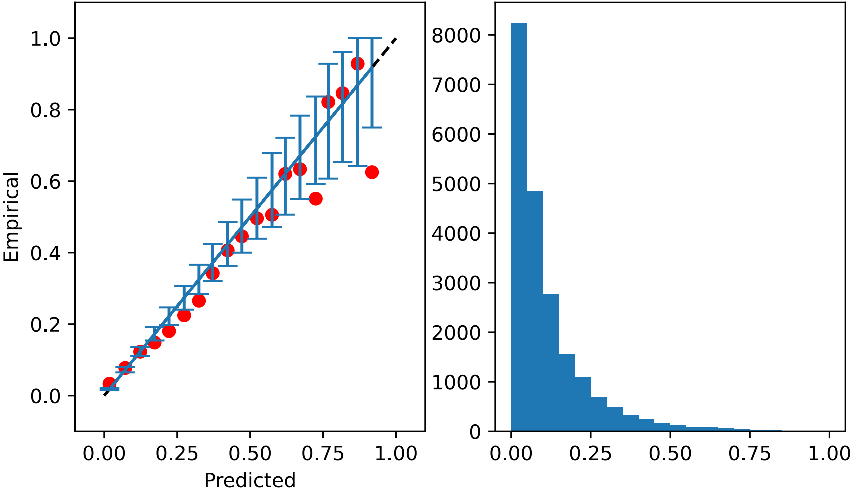
- This figure is from Calibration Workshop Jupyter notebook and uses the plot_reliability_diagram in the ml_insights package.

- 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.

- Note that 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

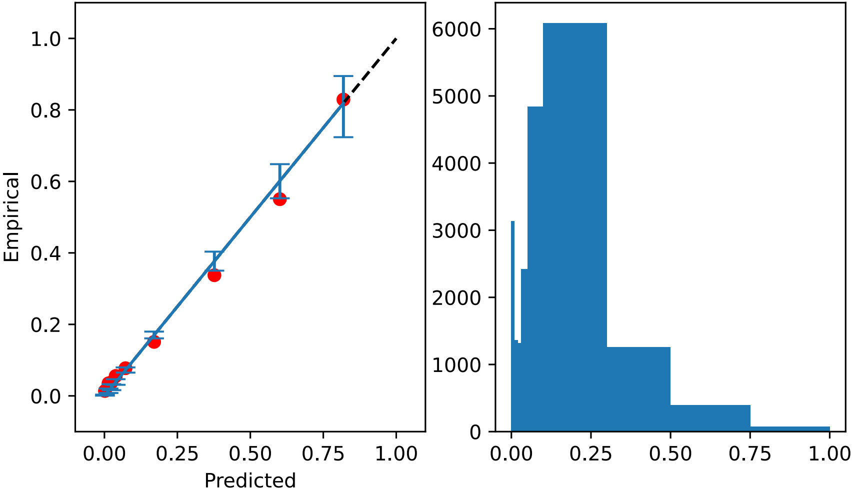
- AUROC: 0.761 Log-loss: 0.388 Square loss: 0.085



- 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. It could also be a serious issue if you're doing something like anomaly detection.
- 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

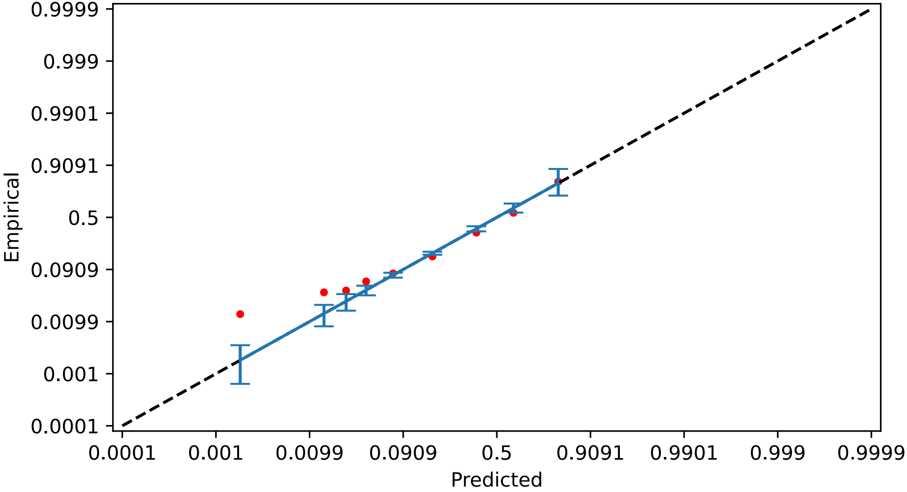
- AUROC: 0.761 Log-loss: 0.388 Square loss: 0.085



- This figure is from Calibration Workshop Jupyter notebook and uses the plot_reliability_diagram in the ml_insights package (bins=np.array([0,.01,.02,.03,.05, .1, .3, .5, .75, 1]), show_histogram=True).
- The highest probability predictions now look calibrated, and issues in the .25 to .75 range are more apparent.
- We're still pretty bunched up for low probabilities though...

Reliability diagram with logit scaling

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



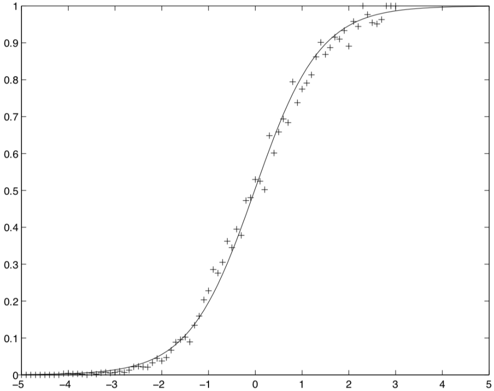
- Now we can see what's going on in the low probability regions better. Calibration doesn't look great down there. The probabilities are consistently lower than they should be for small probabilities, and higher than they should be for probabilities in the middle range.

- For the 3 bins closest to predicted probability of 0.5, the actual probabilities are consistently below the acceptance region. This would indicate that these bins are not calibrated. However, the actual deviation from calibration seems quite small. As practitioners, we have to decide whether or not we care about these small, but significant, deviations. We have a lot of data in these bins, so we may be able to correct the issue using the calibration methods we discuss next.

Calibration methods

Platt scaling (most people's version)

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



[2] Figure 2 from [Pla99]

- The logistic 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. In particular, scikit-learn 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 with label noise

- In [Pla99], Platt is concerned with overfitting a rare class.
- Suppose there are only 3 positive labels, and the data are separable.
- Logistic regression can predict arbitrarily close to probability 1 for those labels.
- Platt's idea is to view each 0/1 label as a noisy version of an actual 0/1 label.
- So for every item x with label 1, we replace it with a probabilistic "soft" label
  - Actual label 1 ↦ Soft label $1-\varepsilon^{+}$
  - Actual label 0 ↦ Soft label ε^{-}
- If $\varepsilon^{+}=\varepsilon^{-}=0.05$, it's essentially equivalent to standard logistic regression where we
  - replace every example $(x,1)$ with 95 examples of $(x,1)$ and 5 examples of $(x,0)$
  - replace every example $(x,0)$ with 95 examples of $(x,0)$ and 5 examples of $(x,1)$
- What should we use for ε^{+} and ε^{-}?

Platt's recommendation for label noise

- Assume label noise is independent of x.
- Suppose we observe N^+ positive labels.
- Assume all N^+ observed positive labels correspond to actual positive labels.
- But this may not always be the case.
- Let's put a uniform prior, i.e. $\text{Beta}(1,1)$, on the probability p that an observed positive label is actually positive.
- Then the posterior on p, assuming all N^+ observed positives are actually positives, is

$$p \sim \text{Beta}(N^+ + 1, 1).$$

- The posterior mean of p is $\frac{N^+ + 1}{N^+ + 2}$.
- Platt recommends replacing positive labels with soft label $\frac{N^+ + 1}{N^+ + 2}$.
- And by the same method, to replace negative labels with soft label $\frac{1}{N^- + 2}$.

- The reasoning on this slide may not be exactly what Platt had in mind. I couldn't exactly follow his reasoning – although if we replace his "MAP" with our "posterior mean", it starts to look much closer.
- I wouldn't call this the most convincing justification.

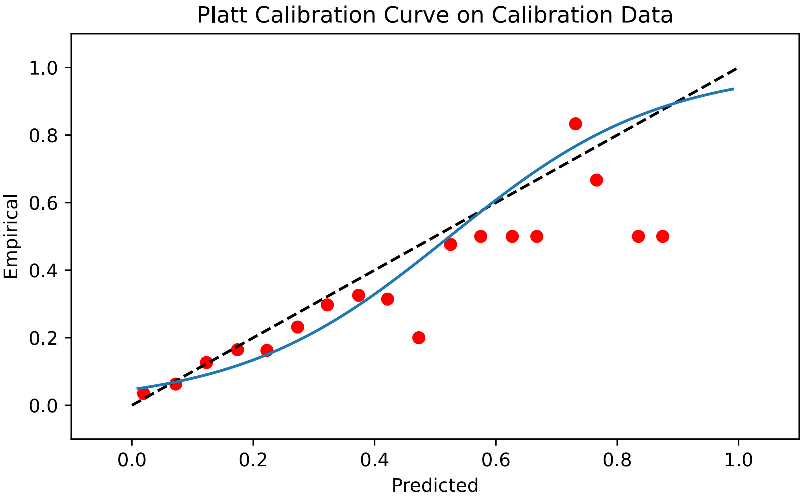
Platt scaling

- Plat scaling assumes a very specific form:

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

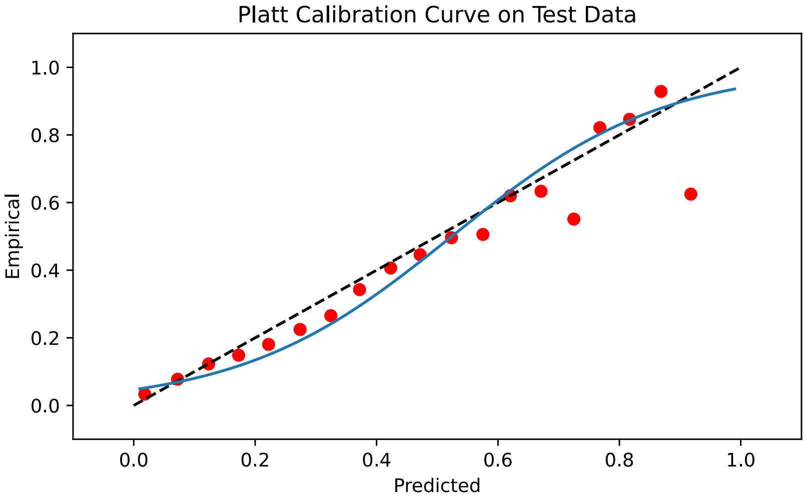
- Advantage: Only 2 parameters, so needs relatively little data to fit.
- Disadvantage: Basically assumes $f(x)$ is on the logit scale.
- If $f(x)$ is a perfectly calibrated probability, Platt scaling will mess it up,
  - since logistic regression does not include the identity function.
  - Unless you're 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



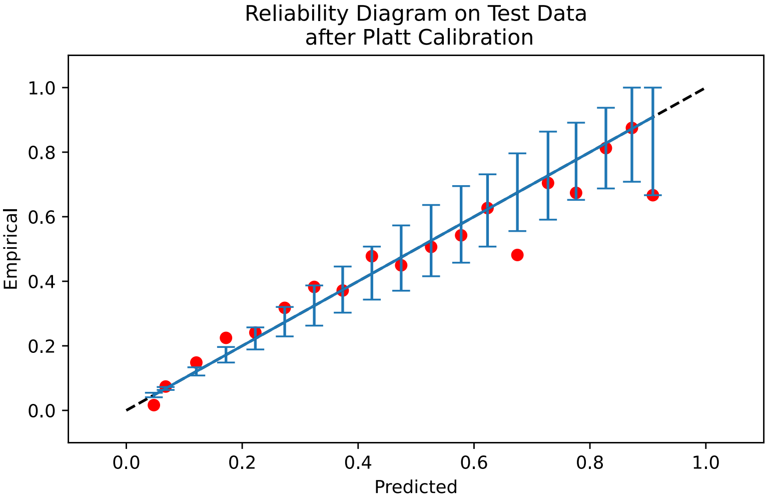
From Brian Lucena's Calibration Workshop Jupyter notebook.

Platt scaling: on test data



From Brian Lucena's Calibration Workshop Jupyter notebook.

Platt scaling: reliability diagram on test data



From Brian Lucena's Calibration Workshop Jupyter notebook.

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
- Results improved for log loss, but actually got worse for Brier score.
- Perhaps the situation would have reversed
  - if we had fit the logistic regression with square loss instead of log-loss.

Other methods

- Isotonic regression: piecewise constant; non-decreasing; needs lots of data [ZE02]
- Binning / histogram method: piecewise constant; needs lots of data
- Spline calibration: smooth, nonparametric [Luc18]
- Generalized additive models: smooth, nonparametric [RSD+12, Sec 3.4]
- Beta calibration: parametric, smooth, somewhat less restrictive than Platt [KFF17]
    - for when original predictions already in $(0, 1)$
- 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 many ML methods are reasonable to try for calibration.

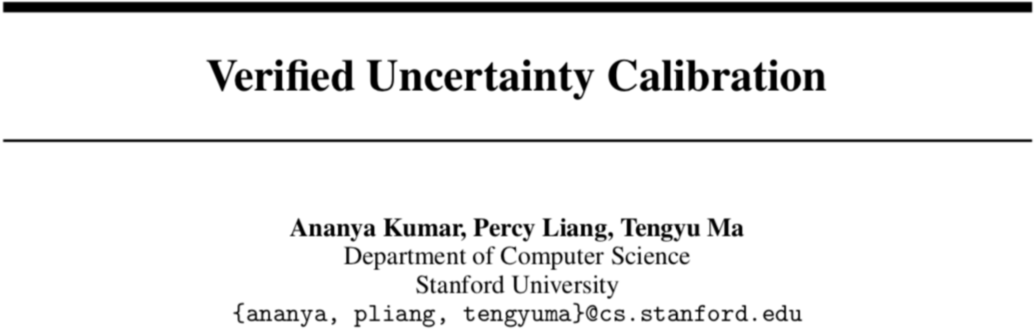
A hack for multiclass

- 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 [ZE02, Sec 5.2].
- How to assess? [KLM19, Sec 2.2]
- **Top-label calibration error**: Compute the CE for the top label of every example
- **Marginal calibration error:** Compute CE for each class and combine with RMS

Formalization of calibration error

Main reference for rest of deck

- 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 with the amount of data we have.
- As we discuss below, we have no way to 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 $\mathbb{P}[Y = 1 \mid f(X)] \approx f(X)$
- "Perfect calibration" as defined above, corresponds to 0 calibration error.
- For better or worse, if $f(x) \equiv \mathbb{E}[Y]$, then $\mathrm{CE}(f) = 0$, i.e. perfect calibration.
- CE is a property of the unknown distribution – we have to **estimate** $\mathrm{CE}(f)$ from data...
- Note: This definition of CE is about squared error. I'm not aware of a similar definition of CE in terms of log-loss. Probably because getting a good estimate of the log-loss for the extreme bins (containing 0 and 1) would require a really large amount of data.

- We can think of $\mathbb{P}[Y=1 \mid f(X)]$ as follows: Consider the collection all x's far which $f(x)$ takes a particular value. $\mathbb{P}[Y=1 \mid f(X)]$ is the probability that $Y=1$, conditioned on selecting an x from the set $\{x : f(x)=f(X)\}$.
- Note the subtle but important change from ISE to CE:

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

- For ISE, we're comparing to the best possible prediction using the information in x. For CE we're comparing to the best possible only using information in $f(x)$.
- So in terms of CE, we can improve our performance by taking a "simpler" f, which throws out more information from X.

What do we 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 (or not, in the case of SVM)...
- Then we mess with it to try to get the calibration better, without messing up the score performance too much.

Decomposition of the Brier score / mean squared error

- Define $t(p) := \mathbb{E}[Y \mid f(X) = p] = \mathbb{P}[Y = 1 \mid f(X) = p]$.
- $t(p)$ is the perfect calibration function for $f(x)$.
- If f is perfectly calibrated, then $t(f(x)) \equiv f(x)$.
- With some math, we can show:

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

- $\mathrm{Var}(t(f(X)))$ is the variance in Y we can account for using the information in $f(X)$.
- If $f(x)$ is perfectly calibrated, so $f(x) = t(f(x))$, then
  - we get a MSE of $\mathrm{Var}(Y) - \mathrm{Var}(t(f(X)))$.
- As $f(x)$ becomes less calibrated, but $t(f(x))$ staying the same, MSE gets worse.
- If $f(x) \equiv \mathbb{E}[Y]$, then calibration error is 0, but sharpness is also 0, so MSE is high.

This form of the decomposition is from [KL15, Sec 2.2], though it seems to be a classical result. There's a version of it in [DF83, Eq. 4.2], but that paper cites a technical report by Tukey et al from 1965, which I can't track down.)

Estimating calibration error

We can define CE, but can we estimate it?

Claim: We cannot estimate CE for a general $f(x) : \mathcal{X} \to [0, 1]$.

- The hard part is estimating $\mathbb{P}(Y = 1 \mid f(X) = \hat{p})$ for all possible $\hat{p}$.
  - For each $\hat{p}$, we need a sample of (X, Y)'s that have $f(X) = \hat{p}$.
- 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?

- A smoothness assumption for $\hat{p} \mapsto \mathbb{P}(Y = 1 \mid f(X) = \hat{p})$ would probably work.
- But I haven't seen this done.
- Another approach is to assume that the set $\{f(x) : x \in \mathcal{X}\}$ is finite.
- This seems to be what's done in practice.

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 mean of all $f(X_i)$ where X_i and x are in the same interval.

Basic approach to estimating calibration of general f

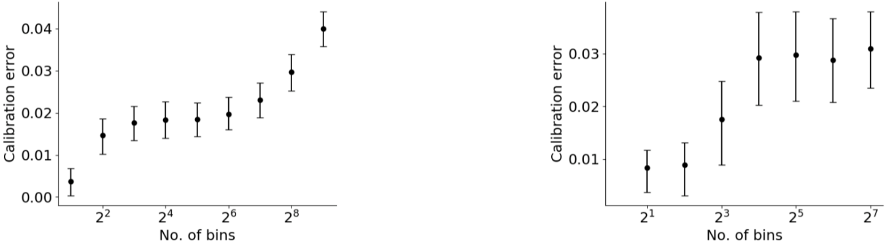
- Choose a binning, usually by equally-sized quantiles
  - so each bin has same amount of data
- For any x, $\hat{f}_{\mathcal{B}}(x) = \text{mean}\{f(X_i) \mid X_i \text{ and } x \text{ are in the same bin}\}$.
- Enumerate the bins $1, \ldots, B$, and let x_b be a representative element of bin b.
- Let $\hat{\mu}_b = \text{mean}\{Y_i \mid X_i \text{ is in bin } b\}$.
- Then $\hat{f}_{\mathcal{B}}(x_b) - \hat{\mu}_b$ is the estimated calibration error for bin b.
- Let $\hat{p}_b = n_b/n$, where n_b is the number of X_i's in bin b.
- The "**plug-in** estimator" of CE is $\hat{\mathcal{E}}_{\text{pl}} = \sqrt{\sum_{b=1}^{B} \hat{p}_b (f_{\mathcal{B}}(x_b) - \hat{\mu}_b)^2}$.

Bias/variance tradeoff in estimating CE

- The SE for the estimate $\hat{\mu}_b$ for any bin is $\sqrt{\hat{\mu}_b(1-\hat{\mu}_b)/n_b}$.
- Fewer bins $\implies$ larger n_b 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.
- But this is just for the estimate of a single bin.
- With more bins, CE is the average of more bin estimates, which should decrease the variance.
- So by one argument, more bins leads to a higher variance CE estimate.
- By another argument, more bins leads to a smaller variance CE estimate.

Empirical tradeoff with number of bins

- From [KLM19, Fig 2]



(a) ImageNet

(b) CIFAR-10

Figure 2: Binned calibration errors of a recalibrated VGG-net model on CIFAR-10 and ImageNet with 90% confidence intervals. The binned calibration error increases as we increase the number of bins. This suggests that binning cannot be reliably used to measure the true calibration error.

\

- Split 50K examples into 3 sets: 20K, 5K, and 25K. 20K set was used to calibrate the model using Platt scaling. Second set was used to compute the quantiles for binning. Third set was used to measure the binned calibration error. 90% confidence intervals computed using 1000 bootstrap samples (not exactly clear which of the 3 sets they bootstrapped). As they used more bins, you can see that the estimated calibration errors increase. Details are in [KLM19, Sec 3.1].

- You can also see that the confidence intervals do not increase substantially. This may be a consequence of what was noted on the previous slide: as we use more bins, we're also averaging together more bin-level CE's to get the overall CE, which would reduce the variance. For the range of bins in these experiments, it seems like these effects essentially cancel out.

Binning underestimates CE

Theorem (Binning underestimates CE)

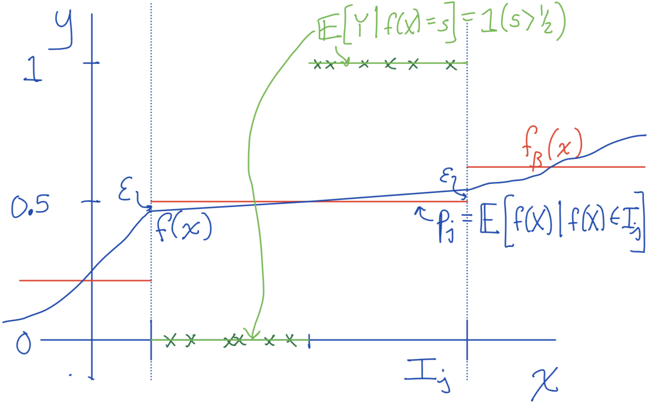
For any binning scheme $\mathcal{B}$ *and model* $f : \mathcal{X} \to [0,1]$ *we have*

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

- Proved in [KLM19, Appendix B].
- Essence of the issue is on the next slide.
- Note: There is no statistical estimation in this theorem. Just approximation of f by $f_{\mathcal{B}}$.

Demonstration of underestimation[3]

- On I_j, $f(x) \approx 0.5$, but actual prob is 0 or 1 $\Longrightarrow$ badly calibrated
- $f_{\mathcal{B}}(x)$ is perfectly calibrated on I_j.



y
1
0.5
0
ε
$f(x)$
$\mathbb{E}[Y \mid f(X)=s] = \mathbb{1}(s > \tfrac{1}{2})$
$f_{\mathcal{B}}(x)$
ε
$p_j = \mathbb{E}[f(X) \mid f(X) \in I_j]$
I_j
x

[3] Example based on [KLM19, Ex. 3.2].

- The key point of this sketch is that $f(x)$ has terrible calibration in the interval I_j, while $f_{\mathcal{B}}(x)$ has perfect calibration in that interval. Yet they are almost identical — what's the difference?
- The key difference is that $f(x)$ takes on a distinct value at every point in the interval. Thus $\mathbb{E}[Y \mid f(X) = s]$ takes a different value for every s. Namely, $\mathbb{E}[Y \mid f(X) = s] = \mathbb{1}[s > 0.5]$. (Where I should have labeled the middle of the intervals as 0.5.) But $f(x)$ is very different from this step function, being about 0.5 away at every x.
- Meanwhile, $f_{\mathcal{B}}(x)$ is constant throughout the interval. So the target for $f_{\mathcal{B}}(x)$ on the interval is $\mathbb{E}[Y \mid f(X) = 0.5] = 0.5$. And $f_{\mathcal{B}}$ is exactly that, so perfectly calibrated.
- So one takeaway is that by taking infinitely many distinct values, it becomes much harder to have good calibration.
- The other major takeaway is that, no matter how good calibration error is on the binned version of f, the actual calibration error of f can be much worse.

Two ways to think about binning

Binning is just a way to estimate CE of f

We like our smooth $f(x)$. We form $f_{\mathcal{B}}(x)$ and estimate its CE. We deploy $f(x)$ and **hope** its CE is not much worse than the 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 the binned version $f_{\mathcal{B}}$ is. 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 of CE defined above is biased.
- The **debiased estimator** for squared calibration error is

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

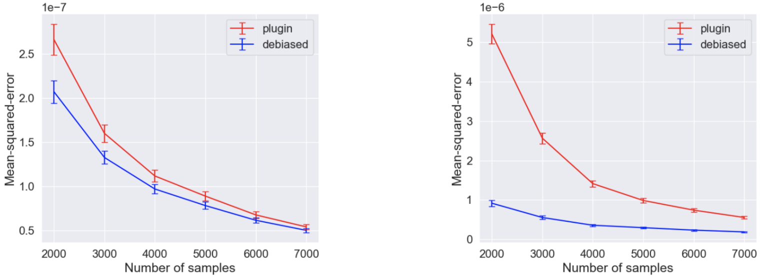
- The main result of [KLM19] is that this estimator is much more sample efficient than the plug-in estimator.

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.

References

Resources

- Brian Lucena's "Probability Calibration Workshop" videos and Jupyter notebooks are recommended.
- [GPSW17] has a lot of interesting experimental results assessing calibration with neural networks using different forms of regularization. Note that they assess calibration error using $B = 15$ bins. [KLM19] notes that when they used 15 bins, they found some of their CE's to be underestimated by at least a factor of two (in the sense that, when they used more bins, they found CE to be twice as high).
- [KPNK$^+$19] uses Dirichlet distributions in a more elegant approach to multiclass calibration.
- [MKS$^+$20] proposes the "focal loss" function, which seems to give more calibrated probabilities in the initial training.

References I

[DF83] Morris H. DeGroot and Stephen E. Fienberg, *The comparison and evaluation of forecasters*, The Statistician **32** (1983), no. 1/2, 12.

[GPSW17] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger, *On calibration of modern neural networks*, Proceedings of the 34th International Conference on Machine Learning - Volume 70, ICML'17, JMLR.org, 2017, pp. 1321–1330.

[KFF17] Meelis Kull, Telmo Silva Filho, and Peter Flach, *Beta calibration: a well-founded and easily implemented improvement on logistic calibration for binary classifiers*, Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (Aarti Singh and Jerry Zhu, eds.), Proceedings of Machine Learning Research, vol. 54, PMLR, 20–22 Apr 2017, pp. 623–631.

References II

[KL15] Volodymyr Kuleshov and Percy S Liang, *Calibrated structured prediction*, Advances in Neural Information Processing Systems 28 (C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, eds.), Curran Associates, Inc., 2015, pp. 3474–3482.

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

[KPNK$^+$19] Meelis Kull, Miquel Perello Nieto, Markus Kängsepp, Telmo Silva Filho, Hao Song, and Peter Flach, *Beyond temperature scaling: Obtaining well-calibrated multi-class probabilities with dirichlet calibration*, Advances in Neural Information Processing Systems 32 (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), Curran Associates, Inc., 2019, pp. 12316–12326.

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

References III

[MKS+20] Jishnu Mukhoti, Viveka Kulharia, Amartya Sanyal, Stuart Golodetz, Philip HS Torr, and Puneet K Dokania, *Calibrating deep neural networks using focal loss*, Advances in Neural Information Processing Systems, 2020.

[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.

[RSD+12] Troy Raeder, Ori Stitelman, Brian Dalessandro, Claudia Perlich, and Foster Provost, *Design principles of massive, robust prediction systems*, Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (New York, NY, USA), KDD '12, Association for Computing Machinery, 2012, pp. 1357–1365.

References IV

[ZE02] Bianca Zadrozny and Charles Elkan, *Transforming classifier scores into accurate multiclass probability estimates*, Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '02, - 2002, pp. 695–699.