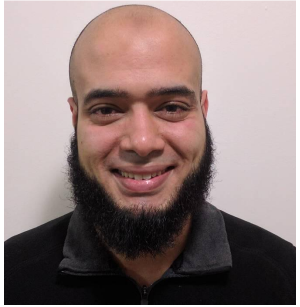
Machine Learning

Model Calibration

Mostafa S. Ibrahim
Teaching, Training and Coaching for more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / MSc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)



When Decision Boundary is enough

- Assume you trained a classifier and used a threshold=0.6 for the **decision boundary**
- Does the final answer (argmax) changes between these 2 answer?
  - No, the largest class anyway is the dog
  - We don't care about the **actual probability** as long as the selected class is right

ML Classifier

- Dog (with 0.8)
- Cat (with 0.2)

- Dog (with 0.6)
- Cat (with 0.4)

Model Calibration

- Model Calibration refers to the process of ensuring that the predicted probabilities of a probabilistic model align with the true outcomes.
- When a calibrated model predicts an event with a probability of **x%,** that event should occur approximately **x%** in the ground truth **at that probability**
  - “Intuitively, you want to have calibration so that you can interpret your estimated probabilities as long-run frequencies”
- When we change the input distribution (e.g. for handling imbalance) or some algorithm's nature (e.g. Random Forests, Gradient Boosted Trees), models may end up being **miscalibrated**!

Model Calibration Status Check

- Assume you trained a spam classifier. Our **val** dataset has 9 examples
  - Spam label = 1. The classifier outputs probability of spam

GT	1	1	0	1	1	1	0	1	0
Pred	0.25	0.25	0.25	0.25	0.25	0.51	0.51	0.51	0.51

- Our classifier predicted 0.25 probability for being spam for **an** example
  - We have 5 examples predicted with similar probability
  - Then we expect 25% of the 5 examples to have gt as spam
  - We see that we have 4 out 5 spam emails, which is = 0.8 ⇒ hence miscalibration
- Our classifier predicted 0.51 probability for an example
  - We have 4 examples predicted with similar probability (2 out of 4 are spam)
  - 0.51 for the classifier vs 2/4 = 0.50 from the ground truth ⇒ very good calibration

When Calibrated Probabilities are a must

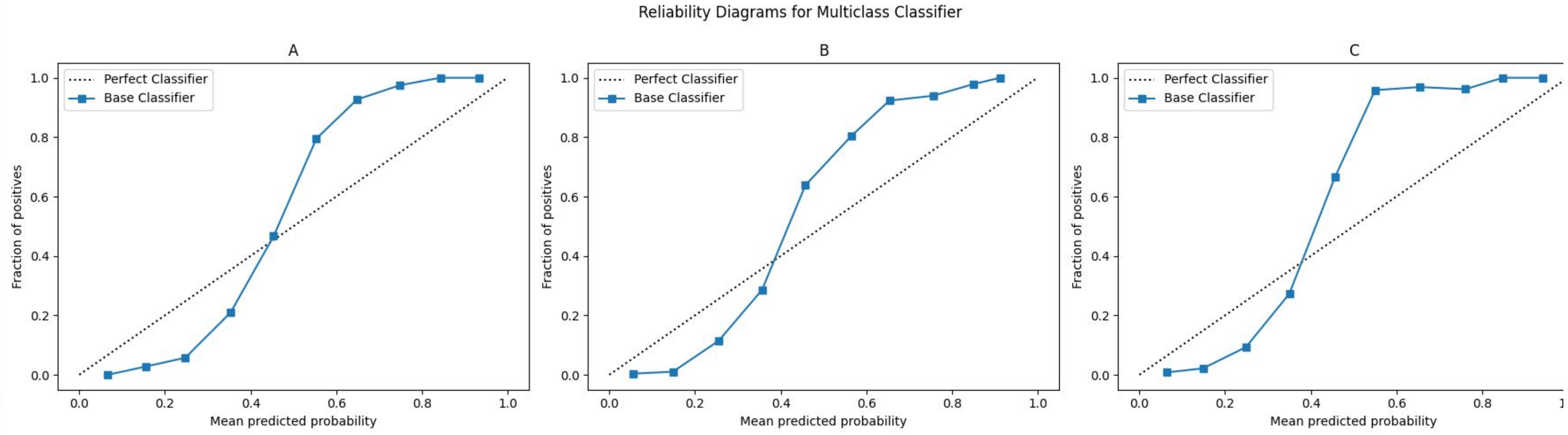
- Imagine Udemy sends **promotional emails** about a new course. A binary classifier to predict whether a user will click on a link in the email
- Before sending to 100k users, we want to know if it worth or not
    - We want to compute the **expected** number of clicks
        - The expected number of clicks for each email is the probability of the click
- Assume for 5 users
    - Miscalibrated classifier probabilities: 0.8, 0.15, 0.1, 0.1, 0.05 sums to 1.2 clicks
    - A calibrated classifier probabilities: 0.6, 0.55, 0.35, 0.25, 0.10 sums to **1.85** clicks
- Wrong probabilities have critical impact (e.g. pay-per-click advertising, video views, # of customer Churn) - In addition to **decisions** based on the probability value (e.g. treatment decision - risk score - earthquakes, evacuation orders)

Visualizing Miscalibration

- Reliability Diagrams (or calibration plots) can be used to visualize miscalibration
    - We will extend the example we gave: instead of finding examples predicted with 0.25, we will find examples with prediction in range [0.20 - 0.29]
    - So we create typically 10 bins dividing probabilities range [0-1] into 10 blocks
    - Calculate for each interval
        - x = Compute the mean of the prediction of examples in this range
        - y = Fraction of Positives from the ground truth
        - Plot point(x, y) and compare with point(x, x) to see the mismatch
            - Ideal Calibration Line: Draw a 45-degree line (from (0,0) to (1,1))
            - Point above line ⇒ under-confident in that interval
            - Point under line ⇒ over-confident in that interval
- All of that can be computed by sklearn.calibration.**calibration_curve**

Visualizing Miscalibration

- I used make_classification to create 5000 examples from 3 classes
  - Data is split to train, val and test
  - We train on a subset and compute the calibration on another
- The 3 curves show miscalibration



```python
X, y = make_classification(n_samples=5000, n_features=20,
                           n_informative=15, n_redundant=5, n_classes=3, random_state=42)

# We need 3 sets: Split the data into train, validation, and test sets
X_train, X_val_test, y_train, y_val_test = train_test_split(X, y, test_size=0.4, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_val_test, y_val_test, test_size=0.5, random_state=42)

clf = RandomForestClassifier(n_estimators=100)
clf.fit(X_train, y_train)

prob_pos_base = clf.predict_proba(X_test)

fig, axes = plt.subplots(nrows=1, ncols=3, figsize=(18, 5))
fig.suptitle('Reliability Diagrams for Multiclass Classifier')

for i, (ax, class_name) in enumerate(zip(axes, ["A", "B", "C"])):
    # Compare reliability diagrams
    fraction_of_positives_base, mean_predicted_value_base = \
        calibration_curve(y_test == i, prob_pos_base[:, i], n_bins=10)

    ax.plot([0, 1], [0, 1], 'k:', label='Perfect Classifier')
    ax.plot(mean_predicted_value_base, fraction_of_positives_base,
            's-', label='Base Classifier')
    ax.set_ylabel('Fraction of positives')
    ax.set_xlabel('Mean predicted probability')
    ax.set_title('%s' % class_name)
    ax.legend()

plt.tight_layout()
plt.subplots_adjust(top=0.85)
plt.show()
```

Enhanced Visualization

- **Histogram**: Alongside the reliability diagram, you can also plot a histogram to show the number of samples in each bin.
  - This provides context regarding where most of your data lies and which intervals might be more reliable due to having more samples.
- **Error Bars**: You can add error bars to each point in the reliability diagram to indicate the uncertainty in the true fraction of positives, especially if some bins have few samples.
- You can use other ways than 10-bins
  - More bins
  - Divide based on percentiles of data - see sklearn

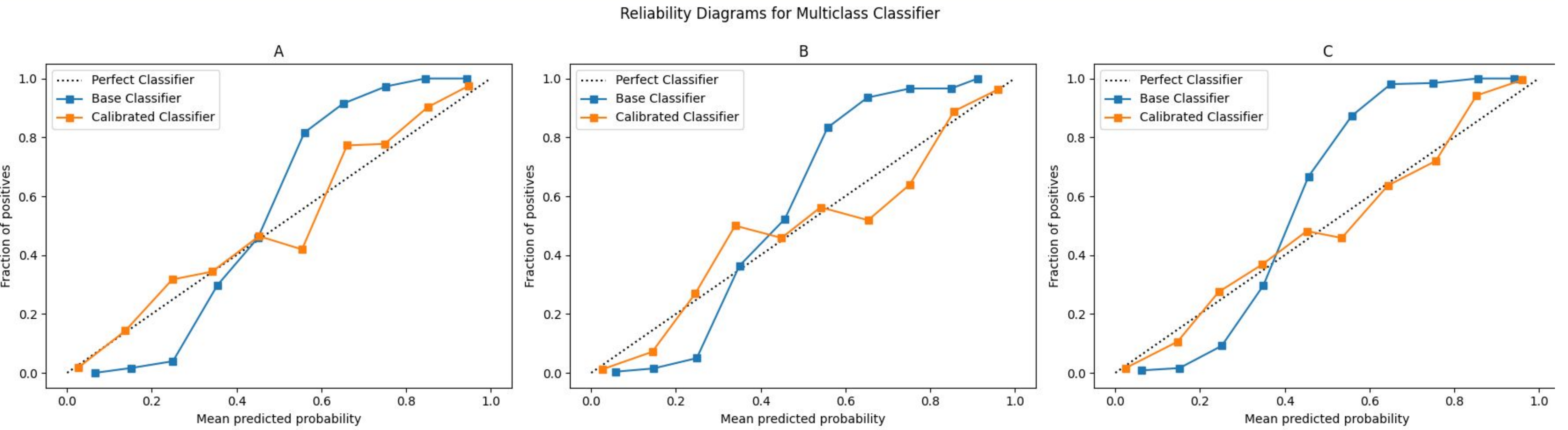
Fixing Miscalibration

- For models that are miscalibrated, **post-processing** calibration techniques can be applied to adjust the output probabilities to be close to a calibrated model
- Examples for techniques
  - Platt scaling - sklearn.calibration. CalibratedClassifierCV with **method='sigmoid'**
    - Fits a logistic regression model to the classifier's scores, transforming them into probabilities
    - Uses a cross-validation generator for val/calibrate sets
    - Extended for multiclass classifier for some classes, e.g. NN
  - Isotonic regression - CalibratedClassifierCV with **method=’isotonic’**
  - Beta calibration and SplineCalib [not on sklearn yet]
- Observe: we first train the model on a train-set, then calibrate on val-set and visualize the curve on a test (or val2)-set (to avoid bias)

Fixing Miscalibration

- Comparing the perfect line with model before and after calibration

```
calibrated_clf = CalibratedClassifierCV(clf, method='sigmoid', cv='prefit')
calibrated_clf.fit(X_val, y_val)
y_pred = calibrated_clf.predict(X_test)
```



Misc

- Many calibration methods are designed to adjust the predicted probabilities so that they are **better aligned** with the true probabilities, but they do not typically change the **relative ranking** of the probabilities.
  - For example, your arg-max class still the same
- "LogisticRegression returns well calibrated predictions by default"
  - GaussianNB and RandomForestClassifier don't
- Seems focal loss helps DNN to be well-calibrated
- In the future when you need calibration, please explore the math behind the different methods / evaluate and compare different methods

Relevant Materials

- Video / Code
- Video / Code
- Why model calibration matters and how to achieve it
- Probabilistic forecasts, calibration and sharpness - paper
- Udemy [no idea of quality]

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."