AMIC Uncertainty – Monte Carlo Method

Leon King

August 2025

1 Introduction

AMIC is a language model that can integrate self-attention neural network models with logistic regression methods to analyze sentiment. However, the current architecture of AMIC can only provide point estimation in either continuous regression or classification. As a statistician, I always want to know the uncertainty level about the point estimation. So this is why we would like to modify AMIC to produce a confidence interval more like a statistic.

2 Methods

There are two major ways to employ uncertainty in the AMIC model. One way is quick and straightforward, but less accurate. It is to add a dropout layer to the AMIC model, then repeat the forward pass of the model to generate a sample of predictions. Finally, the mean, variance, lower quantile, and upper quantile can be calculated from these Monte Carlo samples. Another approach is a more complicated but more accurate method for calculating uncertainty based on the Variational Bayesian. The method is to add one Bayesian layer to the model.

3 Monte Carlo Method

In the class of Synthesizer, which is the final building block of the AMIC model, it has a dropout layer with a probability of 0.2 in the forward pass. After full training, we would run each review on the trained Synthesizer module to generate Monte Carlo samples using this dropout layer.

4 Code Snippet

The following algorithm displays how the samples are drawn and summary statistics are calculated for each review.

Algorithm 1 MC Dropout Inference for AMIC

Require: model with dropout, inputs $(\text{sent}, \text{digits}, \text{mask})$, samples T

Activate dropout at inference: `model.train()`

for $t = 1$ to T **do**

  $\hat{y}^{(t)} \leftarrow \text{model}(\text{sent}, \text{digits}, \text{mask}, \text{use\_mask})$

end for

$\bar{y} \leftarrow \frac{1}{T} \sum_{t=1}^{T} \hat{y}^{(t)}, \quad s^2 \leftarrow \frac{1}{T-1} \sum_{t=1}^{T} (\hat{y}^{(t)} - \bar{y})^2$

return $\{\hat{y}^{(t)}\}_{t=1}^{T}$, $\bar{y}$, s^2

The following code Snippet in Python displays the exact detailed code to realize this simple Monte Carlo Uncertainty. The key argument to this mc_dropout_predict() is the model and T. The model we used here is Synthesizer. Other classes also have dropout layers. Eventually, we can take advantage of each class's

dropout layer to generate samples based on the entire architecture. T refers to the number of samples we want for each observation. Here I set it up to 50, but we can set it up to more samples.

Listing 1 Monte Carlo Dropout prediction wrapper for AMIC/Synthesizer.

```python
def mc_dropout_predict(
    model: torch.nn.Module,
    sent: torch.Tensor,
    digits: torch.Tensor,
    mask: torch.Tensor,
    use_mask: bool = False,
    T: int = 50,
    device: torch.device = None
):
    if device is None:
        device = next(model.parameters()).device
    sent   = sent.to(device)
    digits = digits.to(device)
    mask   = mask.to(device)

    # Enable dropout at inference
    model.train()

    # collect T predictions
    preds = []
    with torch.no_grad():
        for _ in range(T):
            sig_out, _, _ = model(sent, digits, mask, use_mask=use_mask)
            preds.append(sig_out)        # each sig_out: (batch_size,)

    # stack to (T, batch_size)
    preds = torch.stack(preds, dim=0)

    # posterior mean and variance
    mean_pred = preds.mean(dim=0)     # (batch_size,)
    var_pred  = preds.var(dim=0)      # (batch_size,)

    return preds, mean_pred, var_pred
```

Monte Carlo summary statistics. Given a set of T Monte Carlo samples $\{s_t\}_{t=1}^{T}$ from the model's predicted probabilities:

	true_label	model_output	mean_p	var_p	ci_lower	ci_upper	prob1	prob0	mean_label	certain
0	1	0.774113	0.763050	0.001680	0.682938	0.831176	1.00	0.00	1	True
1	1	0.631434	0.628231	0.004422	0.478339	0.724528	0.96	0.04	1	False
2	0	0.693451	0.699905	0.001826	0.618891	0.760362	1.00	0.00	1	True
3	0	0.014944	0.015617	0.000047	0.006947	0.028855	0.00	1.00	0	True
4	0	0.017799	0.021533	0.000195	0.006605	0.044335	0.00	1.00	0	True

Figure 1: Uncertainty Table

$$\text{Posterior mean:} \quad \bar{p} = \frac{1}{T}\sum_{t=1}^{T} s_t, \tag{1}$$

$$\text{Posterior variance:} \quad \sigma_p^2 = \frac{1}{T}\sum_{t=1}^{T} (s_t - \bar{p})^2, \tag{2}$$

$$95\%\text{ credible interval:} \quad \mathrm{CI}_{95\%} = \big[\, Q_{2.5}(s),\ Q_{97.5}(s) \,\big], \tag{3}$$

$$\text{Empirical class probability (class 1):} \quad \hat{P}(y=1) = \frac{1}{T}\sum_{t=1}^{T} \mathbb{I}[\, s_t > 0.5 \,], \tag{4}$$

$$\text{Empirical class probability (class 0):} \quad \hat{P}(y=0) = 1 - \hat{P}(y=1), \tag{5}$$

$$\text{Predicted label (mean threshold):} \quad \hat{y} = \mathbb{I}[\, \bar{p} > 0.5 \,], \tag{6}$$

$$\text{Certainty flag:} \quad \text{certain} = \big(\, Q_{2.5}(s) > 0.5 \,\big) \ \vee\ \big(\, Q_{97.5}(s) < 0.5 \,\big), \tag{7}$$

where $Q_\alpha(s)$ denotes the αth percentile of the sample set and $\mathbb{I}[\cdot]$ is the indicator function.

5 results

The final table from the train dataset would look as follows: If the lower confidence interval limit is greater than 0.5, the probability of being class 1 would be 1. If the upper confidence limit is less than 0.5, the probability of being 0 would be 0. If the 95% confidence interval contains the 0.5, then the frequency of samples greater than 0.5 divided by the total number would be the probability of being 1.

The following violin plot shows that when true labels are 0, the predicted probability of them are more accurate close to 0 than cases of true labels being 1.

The following plot is the confusion matrix. The accuracy from this model is 90.8%. The precision is 84.9%. The recall is 80.9%. The F1 score is 82.9%. The AMIC model is highly accurate overall. The model also has a good balance between precision and recall. The AMIC accuracy without using Monte Carlo samples is 88.42%. However, once we use Monte Carlo samples to calculate the accuracy, it increases to 91%. This means that the uncertainty not only helps to provide more measurements but also improves the point estimation.

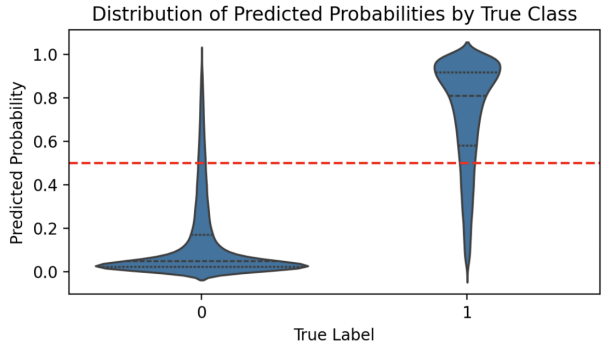

Figure 2: Enter Caption

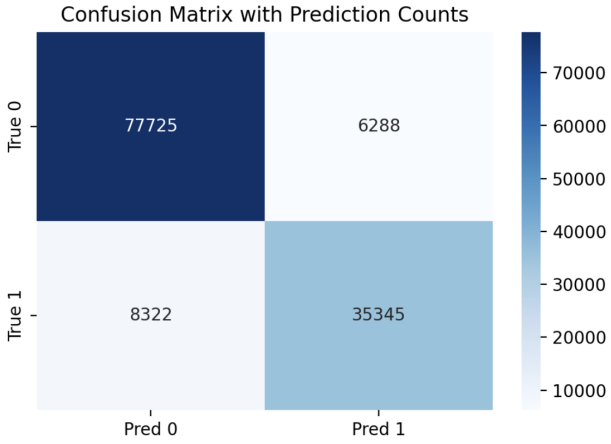

Figure 3: Confusion Matrix