Tuning the Thresholds

As we mentioned in Section , plug-in classifiers learn a CPE model to predict $g(\mathbf{x}) \approx \mathbb{P}[y = +1]$ and then tune a threshold η to obtain a classifier $f(\mathbf{x})\text{sign}(g(\mathbf{x}) - \eta)$. The threshold η is tuned in order to maximize the performance measure of interest, be it classification accuracy or F-measure or G-mean etc. A popular technique to perform this tuning is the Empirical Utility Maximization (EUM) approach (Ye et al. 2012). The EUM approach suggests that a threshold be chosen that maximizes the performance measure on the training dataset.

For simplicity, consider a binary classification problem where we have n labeled data points $(\mathbf{x}_i, y_i)$ where $y_i \in \pm 1$. The CPE model g can be used to obtain CPE scores $s_i = g(\mathbf{x}_i)$. The EUM approach seeks to find a threshold η_{opt} such that the classifier $\text{sign}(g(\mathbf{x}) - \eta_{\text{opt}})$ has, say very high F-measure, on the dataset. However, this search problem is daunting since thresholds are real valued in general. However, other results such as those of (Kotłowski and Dembczynski 2015; Narasimhan, Vaish, and Agarwal 2014) assure us that the optimal threshold will definitely be one of the CPE scores itself i.e. $\eta_{\text{opt}} = s_i$ for some i.

MIML*perf* adopts the same EUM approach in the MIML setting as well. However, there are two key differences here

1. A separate threshold has to be tuned for the plug-in classifier corresponding to all the L labels.
2. The labels for a bag cannot be obtained from the instance CPE scores directly since there is an aggregation step involved.

More specifically, our classifier turns on the label j for a bag i only if that label is turned on for some instance k in that bag. Recall from Algorithm 2 that we set

$$\hat{y}_j = \bigvee_{k=1}^{n_t} \mathbb{I}\left\{ g^j(\mathbf{x}^{(k)}) \geq \eta_j \right\}$$

This presents a challenge since the number of CPE scores to be tuned over can run into millions or more for large datasets due to the large number of bags and large number of instances in each bag. Checking each CPE score as a candidate threshold is thus, not feasible.

We overcome this problem using a simple trick. Notice that the plug-in classifier has a monotonicity property. If $\text{sign}(g(\mathbf{x}) - \eta) = -1$ then $\text{sign}(g(\mathbf{x}') - \eta) = -1$ for all $g(\mathbf{x}') \leq g(\mathbf{x})$. Also, if $\text{sign}(g(\mathbf{x}) - \eta) = +1$ then $\text{sign}(g(\mathbf{x}') - \eta) = +1$ for all $g(\mathbf{x}') \geq g(\mathbf{x})$. Using this, it is easy to see that a label j will be turned on for a bag i if and only if the instance with the largest CPE score for that label in the bag has been assigned that label. More specifically, let $s_{\max}^{(i,j)} := \max_{k \in [n_i]} g^j(\mathbf{x}_i^{(k)})$. Then the following claim holds true

Lemma 2. *For any data point i, label j, CPE model, g^j and threshold η_j, we have $\hat{y}_j = 1$ iff $s_{\max}^{(i,j)} \geq \eta_j$.*

This shows us that while tuning thresholds for the plug-in classifier corresponding to a certain label, it is sufficient to only consider the maximum CPE score for that label in every bag as a candidate threshold. This gives a huge speedup since it decreases the number of thresholds being tuned over from $\sum_{i=1}^{N} n_i$ to simply N. In practice we observe an order of magnitude from this step alone.

However, this is still not sufficient since testing a candidate threshold itself takes $\mathcal{O}(N)$ time since that is the amount of time taken to calculate the performance measure, say F-measure, corresponding to that threshold. This brings the total time taken to tune the L thresholds as $\mathcal{O}\left(N^2 \cdot L\right)$ which is simply infeasible in large scale settings.

Fortunately, the nice structure of the performance measures such as F-micro and F-macro measure come in handy at this point. It turns out that all of these performance measures can be written as some function of the true positive (TP) and true negative (TN) numbers of the classifier being considered. For example, if we look at the label-wise F-measure which is used to define the macro F-measure, we find that if we define

$$\text{TP}^j(\mathbf{f}, \mathbf{X}, \mathbf{Y}) = \sum_{i=1}^{n} \mathbb{I}\left\{ \mathbf{f}^j(\mathbf{x}_i) > 0 \wedge y_{i,j} > 0 \right\}$$

$$\text{FP}^j(\mathbf{f}, \mathbf{X}, \mathbf{Y}) = \sum_{i=1}^{n} \mathbb{I}\left\{ \mathbf{f}^j(\mathbf{x}_i) > 0 \wedge y_{i,j} \leq 0 \right\}$$

$$\text{FN}^j(\mathbf{f}, \mathbf{X}, \mathbf{Y}) = \sum_{i=1}^{n} \mathbb{I}\left\{ \mathbf{f}^j(\mathbf{x}_i) \leq 0 \wedge y_{i,j} > 0 \right\},$$

then we can write $F_\beta^j(\mathbf{f}; \mathbf{X}, \mathbf{Y})$ as

$$\frac{\text{TP}^j(\mathbf{f}, \mathbf{X}, \mathbf{Y})}{\text{TP}^j(\mathbf{f}, \mathbf{X}, \mathbf{Y}) + \beta \cdot \text{FP}^j(\mathbf{f}, \mathbf{X}, \mathbf{Y}) + (1 - \beta) \cdot \text{FN}^j(\mathbf{f}, \mathbf{X}, \mathbf{Y})}.$$

Similar expressions can be obtained for a large family of performance measures (Koyejo et al. 2014; Narasimhan, Vaish, and Agarwal 2014; Narasimhan, Kar, and Jain 2015). These expressions are very useful since the terms TP^j, FP^j etc can be calculated for *all* candidate thresholds in time $\mathcal{O}(N \log N)$. We simply need to sort all the candidate CPE scores in increasing order and simply do a linear scan to find out what value of TP^j, FP^j etc does every candidate threshold offer. This brings down the total time taken to tune the L thresholds from $\mathcal{O}\left(N^2 \cdot L\right)$ to $\mathcal{O}(N \log N \cdot L)$.

We conclude this section by noting that while optimizing macro F-measure, we tune a different threshold per label, whereas we tune a single threshold while optimizing micro F-measure.

Estimating the Hidden Variables

As mentioned in Section , MIML*perf* uses the trained CPE models to obtain CPE scores for every instance with respect to every label. Thus, the method now has an estimate $g^j(\mathbf{x}_i^{(k)})$ of the probability that the kth instance in bag i expresses label j. These are now used to reassign the hidden variables. As discussed, we do not wish to trust these scores completely as they are noisy. Thus, we do not wish to set $z_k^{(i,j)} = 1$ only for the instances k with the highest values of $g^j(\mathbf{x}_i^{(k)})$. Instead we *sample* instances the instances according to their CPE scores to reassign them labels.

More specifically, if a bag i has a label j (recall that if the bag does not have label j then we can simply set $\mathbf{z}^{(i,j)} = \mathbf{0}$ since no instance in that bag could be expressing label j) then we sample the k^{th} instance with probability $g^j(\mathbf{x}_i^{(k)}) / \sum_{k'=1}^{n_i} g^j(\mathbf{x}_i^{(k')})$ where n_i is the number of instances in bag i. Using this procedure we create a set $S^{(i,j)}$ of $c^{(i,j)}$ instances. We now simply set $z_k^{(i,j)} = 1$ if $k \in S$ and $z_k^{(i,j)} = 0$ otherwise.

The choice of $c^{(i,j)}$ can be varied. In our experiments, we used $c^{(i,j)} = \kappa \cdot n_i$ in the initialization step where κ is the expression rate parameter. In subsequent steps, we used $c^{(i,j)} = \sum_{k=1}^{n_i} \mathbb{I}\{g^j(\mathbf{x}_i^{(k)}) \geq \eta_j\}$, i.e. we ask the plug-in classifier trained in the previous iteration, how many instances in that bag were found to express a certain label.