KL-λ-optimal post model threshold search

Hilding Wollbo

1 KL-Divergence in Machine Learning

The Kullback-Leibler (KL) Divergence is an information theoretic concept used across many different probabilistic domains, including the field of Machine Learning. The KL-divergence is defined as

$$\mathbf{KL}[p||q] = -\int p(x) \log \frac{q(x)}{p(x)} dx \tag{1}$$

and can be thought of as a metric describing the distance between two probability distributions p and q. The KL-divergence is however not a true distance metric since it is neither symmetric in that $\mathbf{KL}[p||q] \neq \mathbf{KL}[q||p]$ nor does it satisfy the triangle inequality.

In the case of logistic regression, one tries to fit model parameters φ such that the computed probability $\hat{P}_\varphi(y = 1|x)$ is as close to the true probability $P(y = 1|x)$ as possible, for every point x in the dataset. The difference between these two distributions can be quantified by their KL-divergence, and the problem can be reduced to finding the set of parameters φ that minimize the average KL-distance of the dataset such that

$$\varphi : \arg\min_{\varphi} \Big\{ -\frac{1}{N} \sum_{n=1}^{N} P(y_n = 1|X_n) \log \frac{\hat{P}_\varphi(y_n = 1|X_n)}{P(y_n = 1|X_n)} + P(y_n = 0|X_n) \log \frac{\hat{P}_\varphi(y_n = 0|X_n)}{P(y_n = 0|X_n)} \Big\} \tag{2}$$

Noting that the minimization is performed with regard to φ one can disregard the independent terms in the optimization, which leads to the equivalent problem of minimizing the average cross entropy loss

$$\varphi : \arg\min_{\varphi} \Big\{ -\frac{1}{N} \sum_{n=1}^{N} P(y_n = 1|X_n) \log \hat{P}_\varphi(y_n = 1|X_n) + P(y_n = 0|X_n) \log \hat{P}_\varphi(y_n = 0|X_n) \Big\} \tag{3}$$

This is also equivalent to the Maximum Likelihood formulation of finding the model parameters φ which maximizes the probability that our model predictions $\hat{\mathbf{y}}$ gave rise to the set of observed true labels $\mathbf{y}$.

2 Post model threshold selection

In the case of binary classification, the trained model outputs a probability $f_\varphi(x) \in [0, 1]$ for each input x which is then thresholded to either 0 or 1 by a threshold θ. The default threshold is generally set to $\theta = 0.5$, which for well behaved and balanced datasets can be sufficient. However, the performance of the model predictions on the validation dataset can often be improved by shifting the threshold by maximizing a set of relevant metrics such as precision, recall, F_1-measure etc. depending on application. This view of threshold selection is generally to maximize the average performance of the model output for a specified metric M by varying the threshold θ:

$$\theta : \arg\max_{\theta} \{ \frac{1}{N} \sum_{n=1}^{N} M(\theta(f_\varphi(x_n)), y_n \} \tag{4}$$

or equivalently, to minimize an expected total cost associated with the respective errors as

$$\theta : \arg\min_{\theta} \{ \frac{1}{N} \sum_{n=1}^{N} \alpha y_n (1 - \theta(f_\varphi(x_n))) + \beta(1 - y_n)\theta(f_\varphi(x_n)) \} \tag{5}$$

where α is the cost of a false negative and β that of a false positive. However, as given by the prediction problem we have no way of knowing the true labels y_n. We could use the training data to create predictions and calculate an expected cost and select an optimal threshold on this data. Still, we would like to incorporate the information about the test predictions in our threshold selection. Instead, given that we can collect a sufficient set of unthresholded test predictions, we can use a probabilistic view and just consider the relation between the ideal predictor $\hat{y}^*$ and the true data labels y. One property that an ideal predictor must fulfill is that $\hat{y}^* \sim p(y)$, since all predictions are correct. That is, the proportions of each class in the predictions should equal those in the dataset labels. This essentially means that, instead of minimizing the expected total error of our predictions, we could try to find an optimal threshold θ^* after training by minimizing the distance between the distribution of true class labels $p(y)$ and the global distribution of thresholded predictions $q_\theta(\hat{y})$.

3 Binary classification

Binary classification is the most simple application of prediction in machine learning, but also the most fundamental, since every classification problem can be formulated as sets of binary prediction tasks or decision trees. For a given post model classification problem we have a set of true labels following a distribution $p(y)$ which can be estimated empirically directly from the training data (assuming that both training and test data follow the same distribution). Likewise for the predictions $\hat{y}$ we can define an empirical distribution given a threshold θ as

$$q_\theta(\hat{y}) = \begin{cases} 1, & \text{w. p. } \frac{1}{N}\sum_{n=1}^{N}\theta(\hat{y}_n) \\ 0, & \text{w. p. } 1 - \frac{1}{N}\sum_{n=1}^{N}\theta(\hat{y}_n) \end{cases}$$

, where $\theta(\hat{y}_n) = 1$ if $\hat{y}_n \geq \theta$, else 0. In the case of discrete class classification, the integral in the KL-divergence is replaced with a sum, such that

$$\mathbf{KL}[p(y)||q_\theta(\hat{y})] = -\sum p(y)\log\frac{q_\theta(\hat{y})}{p(y)}, \tag{6}$$

and for the binary classification problem we simply insert our binary probabilities

$$\begin{aligned} \mathbf{KL}[p(y)||q_\theta(\hat{y})] = & -p(y=1)\log\frac{q_\theta(\hat{y}=1)}{p(y=1)} - p(y=0)\log\frac{q_\theta(\hat{y}=0)}{p(y=0)} & (7) \\ = & -p\log\underbrace{\frac{P(\hat{y}\geq\theta)}{p}}_{\text{"fnr cost"}} - (1-p)\log\underbrace{\frac{P(\hat{y}<\theta)}{1-p}}_{\text{"fpr cost"}}. & (8) \end{aligned}$$

In the binary case, each term is associated with an error cost depending on the amount of respective errors resulting from a given threshold, similar to the expected cost in Equation (5). With a too conservative threshold the probability of a negative in the global distribution of predictions $q_\theta(\hat{y}=0)$ is larger than that of the true negative labels $p(y=0)$, resulting in an increased amount of false negatives. This in turn implies that the proportion of predicted positives must be smaller than the true amount of positives, $q_\theta(\hat{y}=1) < p(y=1)$. That is, the classifications follow

$$q_\theta(\hat{y}=0) > p(y=0) \iff TN + FN > \underbrace{TN+FP}_{N} \rightarrow FN > FP.$$

Vice versa holds for the case when $q_\theta(\hat{y}=1) > p(y=1)$, where the proportion of predicted positives is larger than that of the true distribution. However, the cost associated with a false positive error is generally different than that of a false negative, this relation can be captured by a constant λ. One may then define the threshold optimization problem as:

$$\theta : \arg\min_\theta \Big\{ -p\log\frac{P(\hat{y}\geq\theta)}{p} - \lambda(1-p)\log\frac{P(\hat{y}<\theta)}{1-p} \Big\} \tag{9}$$

For $\lambda > 1$, false positives are associated with a higher cost during minimization (in relation to their prevalence in the true labels). This causes the resulting proportion of predicted positives given the threshold to be smaller. Again, the optimization is performed with regard to θ and the independent

terms can be disregarded, leading to the equivalent optimization problem of minimizing the weighted cross entropy loss between the two distributions as

$$\theta : \arg\min_{\theta} \Big\{ -p \log P(\hat{y} \geq \theta) - \lambda(1-p) \log P(\hat{y} < \theta) \Big\}. \tag{10}$$

For brevity, we set $q = P(\hat{y} \geq \theta)$ and the minimization can be expressed as

$$\begin{aligned}
\theta : \arg\min_{\theta} \Big\{ -p \log q - \lambda(1-p) \log(1-q) \Big\} = \\
= \arg\min_{\theta} \Big\{ -\log q^p - \log(1-q)^{\lambda(1-p)} \Big\} \\
= \arg\min_{\theta} \Big\{ -\log q^p (1-q)^{\lambda(1-p)} \Big\} \\
= \arg\min_{\theta} \Big\{ -q^p (1-q)^{\lambda(1-p)} \Big\}.
\end{aligned}$$

Deriving and solving for zero, we have the expression

$$\begin{aligned}
\partial_q q^p (1-q)^{\lambda(1-p)} &= -\lambda(1-p)(1-q)^{\lambda(1-p)-1} q^p + p q^{p-1} (1-q)^{\lambda(1-p)} \\
&= (1-q)^{\lambda(1-p)-1} (-\lambda(1-p) q^p + p(1-q) q^{p-1}) \\
&= 0
\end{aligned}$$

leading to the closed form solution

$$p(1-q)q^{p-1} = \lambda(1-p)q^p \tag{11}$$

$$\rightarrow q = \frac{p}{p + \lambda(1-p)}. \tag{12}$$

Here, q corresponds to a resulting proportion of positive predictions associated with a certain threshold θ when applied to the raw predictions. This threshold can be found by finding the index m in the sorted predictions which corresponds to the proportion q of resulting positively classified predictions, according to

$$m : \arg\min_{m} \Big\{ \Big| q - \frac{1}{N} \sum_{n=m}^{N} \delta(f_\varphi(x_n)) \Big| \Big\}, \; m \in [0, N] \tag{13}$$

$$\theta = f_\varphi(x_m) \tag{14}$$

In this way, we arrive at a structured method of quantifying the tradeoff between probability of errors and the associated costs of each error to find the optimal threshold in the binary prediction setting.

3.1 Example

We may also find specific solutions to this threshold optimization problem in cases where the distribution of test predictions is known but the actual test predictions are not. Given that the set of predictions is modelled by the following, exponential mixture distribution

$$\begin{aligned}
f_0(x|y=0;\beta_0) &= \frac{\beta_0}{Z_0} e^{-\beta_0 x} = \frac{\beta_0}{1 - e^{-\beta_0}} e^{-\beta_0 x} \\
f_1(x|y=1;\beta_1) &= \frac{\beta_1}{Z_1} e^{\beta_1 x} = \frac{-\beta_1}{1 - e^{\beta_1}} e^{\beta_1 x} \\
f(x;\beta_0,\beta_1,\alpha) &= (1-\alpha) \frac{\beta_0}{1 - e^{-\beta_0}} e^{-\beta_0 x} - \alpha \frac{\beta_1}{1 - e^{\beta_1}} e^{\beta_1 x}
\end{aligned}$$

we can find the optimal threshold θ by finding the point where the proportion of expected predicted positives is equal to the one found in Equation (11). We recognize that $\alpha = p$ and set

$$\frac{p}{p + \lambda(1-p)} = \int_{\theta}^{1} f(x;\beta_0,\beta_1,p) dx = \tag{15}$$

$$= \int_{\theta}^{1} \frac{1-p}{1 - e^{-\beta_0}} \beta_0 e^{-\beta_0 x} - \frac{p}{1 - e^{\beta_1}} \beta_1 e^{\beta_1 x} dx = \tag{16}$$

$$= \frac{1-p}{1 - e^{-\beta_0}} (e^{-\beta_0} - e^{-\beta_0 \theta}) - \frac{p}{1 - e^{\beta_1}} (e^{\beta_1} - e^{\beta_1 \theta}). \tag{17}$$

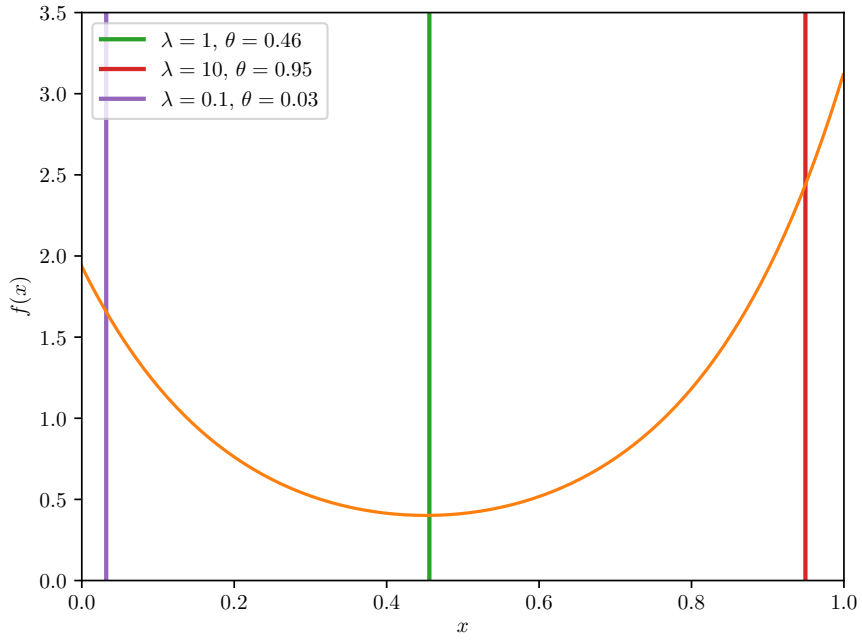

Figure 1: Thresholds for an exponential test prediction distribution with parameter $\beta = 5$ in a dataset of 61 % positive examples.

Rearranging and setting $\beta_1 = \beta_0 = \beta$ for simplicity, we can define the function

$$f(\theta) = \frac{e^{-\beta\theta}(e^{\beta} - e^{\beta\theta})(p(e^{\beta\theta} - 1) + 1)}{e^{\beta} - 1} - \frac{p}{p + \lambda(1-p)} = 0 \tag{18}$$

$$f'(\theta) = \frac{\beta e^{-\beta\theta}((p-1)e^{\beta} - pe^{2\beta\theta})}{e^{\beta} - 1}. \tag{19}$$

Using Newton's Method we can find an approximate solution to θ by iterating

$$\theta_{n+1} = \theta_n - \frac{f(\theta_n)}{f'(\theta_n)} \tag{20}$$

until convergence. An example of this method is shown in Figure 1.

3.2 Statistical significance

The whole framework henceforth presented assumes that we have a batch of test predictions from which we are able to infer a threshold representative of the true distribution of test predictions. However, if by chance, we end up with a set of test predictions with class balance widely different from the training data, we can be sure to arrive at suboptimal thresholds. Therefore, in relation to the class balance, we need to define criteria for which we either proceed with calculating the threshold or simply wait for more data to become available before proceeding.

Assuming that the number of test predictions in the set is sufficiently large (i.e. $pN \geq 10, (1-p)N \geq 10$) to allow us to view q as normally distributed around the true proportion $\bar{q}$, we can use a z-test to determine how many samples we need to achieve at most error $E = q - \bar{q}$ given a certain confidence level α. The test statistic for the z-test is calculated by

$$z_{\alpha/2} = \frac{q - \bar{q}}{\frac{\sigma_q}{\sqrt{N}}}. \tag{21}$$

From here, we may set the confidence interval and error tolerance and solve for N. We already assumed that the proportion of positives in the training data is roughly the same as in the test data,

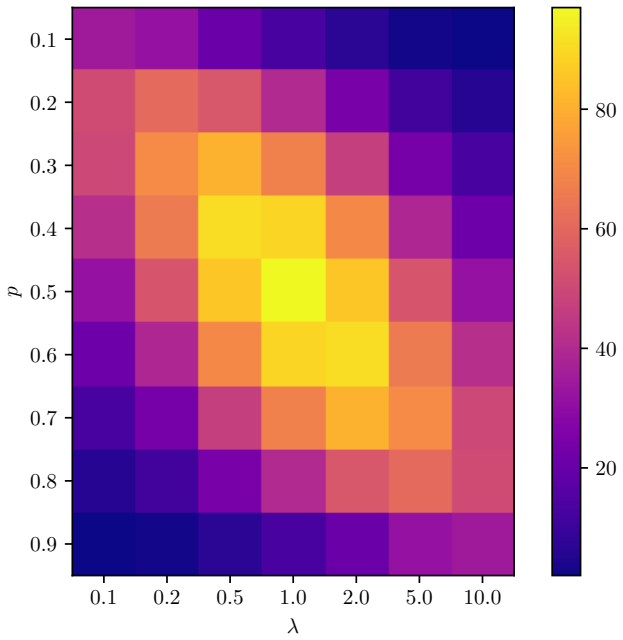

Figure 2: Heatmap of the required number of samples N to achieve at most an error of 0.05 with a 95% CI when estimating q.

so it is justified that we calculate the population variance from the training data as

$$\begin{aligned}\sigma_q &= \sqrt{q(1-q)} \\ &= \frac{1}{p+\lambda(1-p)}\sqrt{\lambda p(1-p)} \\ &= \frac{\sqrt{\lambda}}{p+\lambda(1-p)}\sigma_p\end{aligned}$$

Furthermore, we need to decide a margin of error and a confidence interval for our estimation. Since our threshold θ and proportions p and q all lie in the interval $[0,1]$, it can be suitable to assume that we want an error no larger than 0.05 from the true weighted proportion $\bar{q}$. Following standard practice, we also select a 95% confidence interval. Using the two-sided Z-test, we have

$$E = z_{\alpha/2}\frac{\sigma_q}{\sqrt{N}} \tag{22}$$

Thus, to find how large N needs to be for our requirements to hold, we calculate

$$\begin{aligned}N &= (\frac{z_{\alpha/2}}{E})^2\sigma_q^2 \\ &= (\frac{z_{\alpha/2}}{E})^2\frac{\lambda}{(p+\lambda(1-p))^2}\sigma_p^2\end{aligned}$$

For the chosen values, Figure 2 shows a heatmap of the number of samples N needed to confidently use the developed threshold selection scheme.

4 Practical examples

In order to illustrate how this threshold search can be used in practice, a few simple cases are studied for known binary datasets.

4.1 Setup

In order to generate predictions, a small decision tree model using LightGBM was implemented with default parameters. The project was implemented in Python using the `LGBMClassifier` package,

as well as `sklearn`, `numpy` and `pandas` for data manipulation. The code used to produce the results is available at `https://github.com/wollbo/threshold`. The labels of the dataset are marked accordingly: positive test samples in blue, and negative test samples are marked in orange. The probability output from the classifier is presented as histograms, with several thresholds θ calculated from the classifier predictions using Equation (11) for different misclassification cost ratios λ. Points to the right of a threshold is classified as positive, data to the left of the threshold is classified as negatives.

4.2 Breast Cancer Wisconsin

The Breast Cancer Wisconsin dataset is a traditional dataset used for evaluating baseline performance for binary classifiers, containing a majority positive samples. It is comprised of features computed from digitized images of mammograms. In this case, the simple decision tree model is able to almost perfectly separate the positive and negative samples with only a few outliers. Notably, the raw output probabilities are also very distinctly 1 or 0.

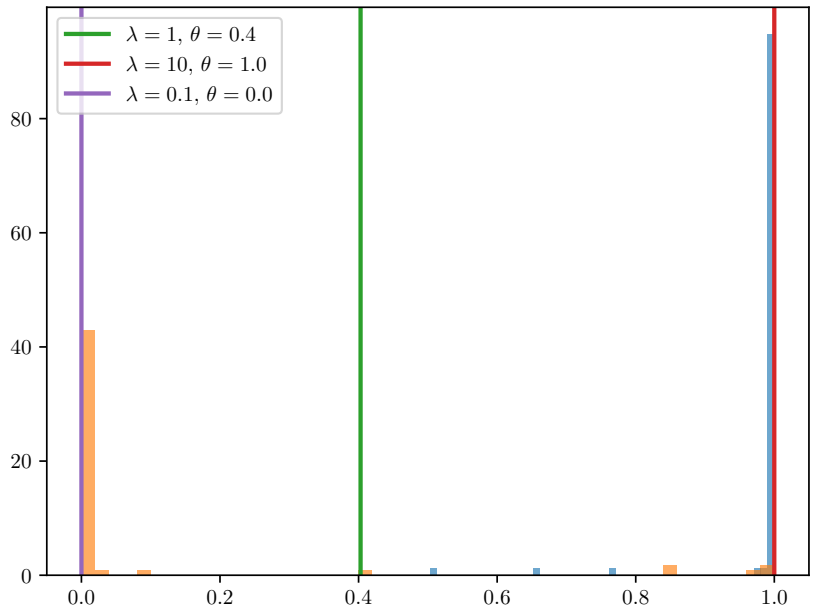

Figure 3: Thresholds and predictions generated for the Wisconsin Breast Cancer dataset, 64 % positive samples. This is a very easy dataset to classify.

4.3 German Credit Data

The German Credit dataset is comprised of several categorical and numerical attributes associated with the financial status of individuals as to determine their credit worthiness, with the majority of samples being positive (corresponding to a 'good' credit status). This is also a rather traditional dataset, but is considerably harder to classify than the Breast Cancer dataset. Still, the algorithm is generally able to separate the positives and negatives, and the threshold selection shows how increasing the parameter λ increases the propensity to classify samples as negatives, and how lowering λ results in classifying most samples as positives.

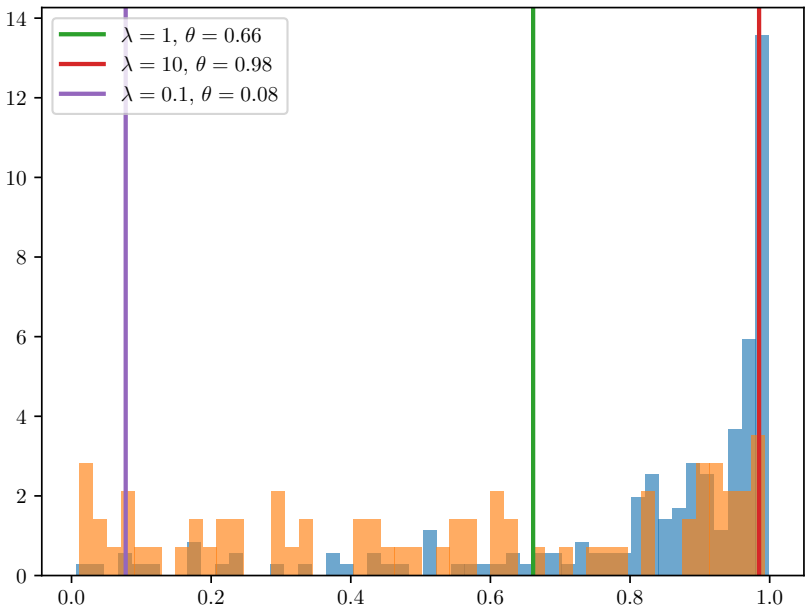

Figure 4: Thresholds and predictions generated for the German Credit Data dataset, 71 % positive samples.

4.4 KDD CUP 2009 - Churn

The KDD CUP 2009 dataset was initially used for the competition bearing the same name, and consists of customer relationship data of the French Telecom company Orange. In this dataset, each customer has 190 numerical and 40 categorical features which are very sparse. Corresponding to the feature data is the labels of 'Churn' which is used to train and evaluate the model. This is a much harder dataset to classify than the other two, and the decision tree model struggles to separate the two distributions. In this case, only 7% of the samples are positive, which means that the selected thresholds become shifted to 0.

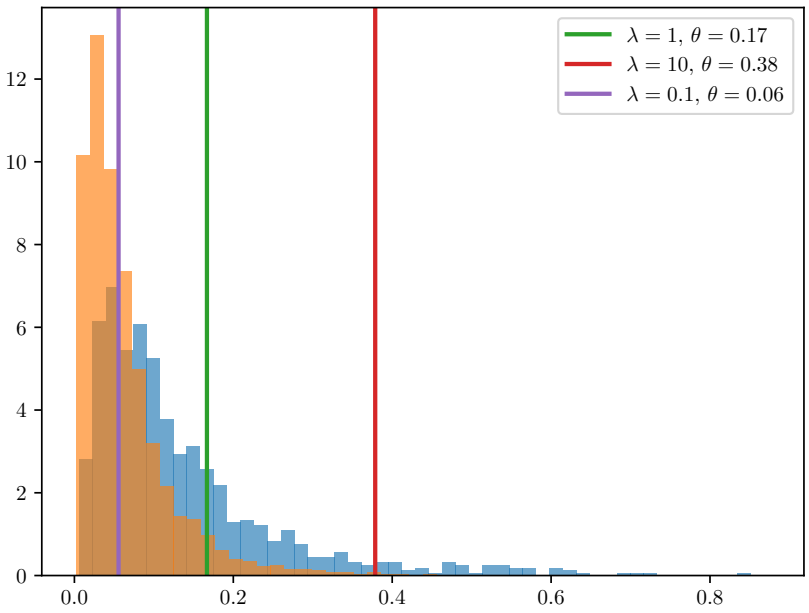

Figure 5: Thresholds and predictions generated for the KDD Cup 2009 Churn dataset, 7 % positive samples. This is a hard dataset to classify.