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

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

where $\theta(\hat{y}_i) = 1$ if $\hat{y}_i \geq \theta$, 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, if

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

Vice versa holds for the case when $P(\hat{y}=1) > p(y=1)$, 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). Again, the optimization is performed with regard to θ, so 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

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

. Deriving and solving for zero, we have the expression

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

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. In this way, we arrive at a structured way of quantifying and minimizing the tradeoff between probability of errors and the associated costs of each error.

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 12. We recognize that $\alpha = p$ and set

$$\frac{p}{p+\lambda(1-p)} = \int_{\theta}^{1} f(x;\beta_0,\beta_1,p)dx = \tag{13}$$

$$= \int_{\theta}^{1} \frac{1-p}{1-e^{-\beta_0}}\beta_0 e^{-\beta_0 x} - \frac{p}{1-e^{\beta_1}}\beta_1 e^{\beta_1 x}dx = \tag{14}$$

$$= \frac{1-p}{1-e^{-\beta_0}}(e^{-\beta_0} - e^{-\beta_0\theta}) - \frac{p}{1-e^{\beta_1}}(e^{\beta_1} - e^{\beta_1\theta}) \tag{15}$$

.

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

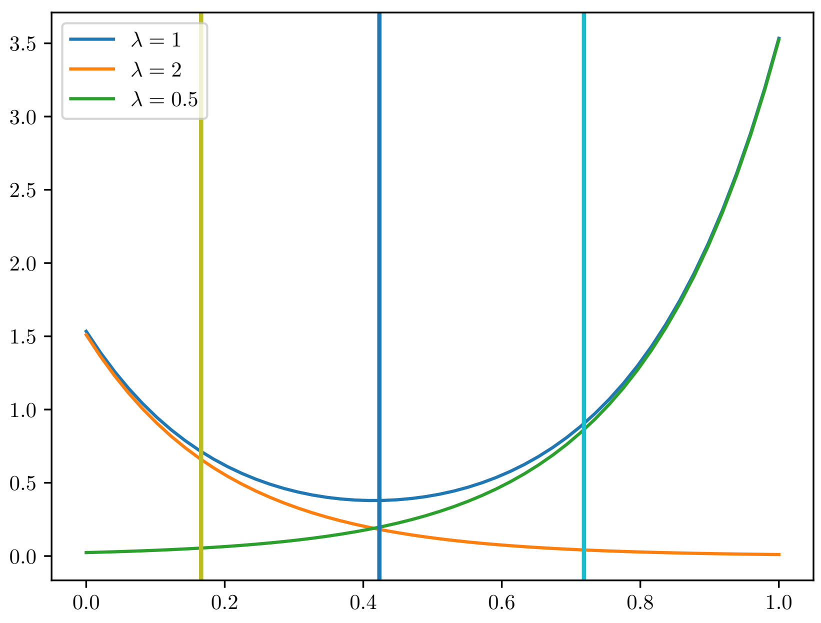

Figure 1: Thresholds for an exponential test prediction distribution with parameter $\beta = 5$ in a dataset of 70 % positive examples.

$$f(\theta) = \frac{e^{-\beta\theta}(e^{\beta} - e^{\beta\theta})(p(e^{\beta\theta} - 1) + 1)}{e^{\beta} - 1} - \frac{p}{p + \lambda(1-p)} = 0 \tag{16}$$

$$f'(\theta) = \frac{\beta e^{-\beta\theta}((p-1)e^{\beta} - pe^{2\beta\theta})}{e^{\beta} - 1} \tag{17}$$

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

$$\theta_{n+1} = \theta_n - \frac{f(\theta_n)}{f'(\theta_n)} \tag{18}$$

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

4 Practical examples

In order to illustrate how this threshold search can be used, 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`.

4.2 Breast Cancer Wisconsin

4.3 German Credit Data

4.4 KDD CUP 2009 - Churn

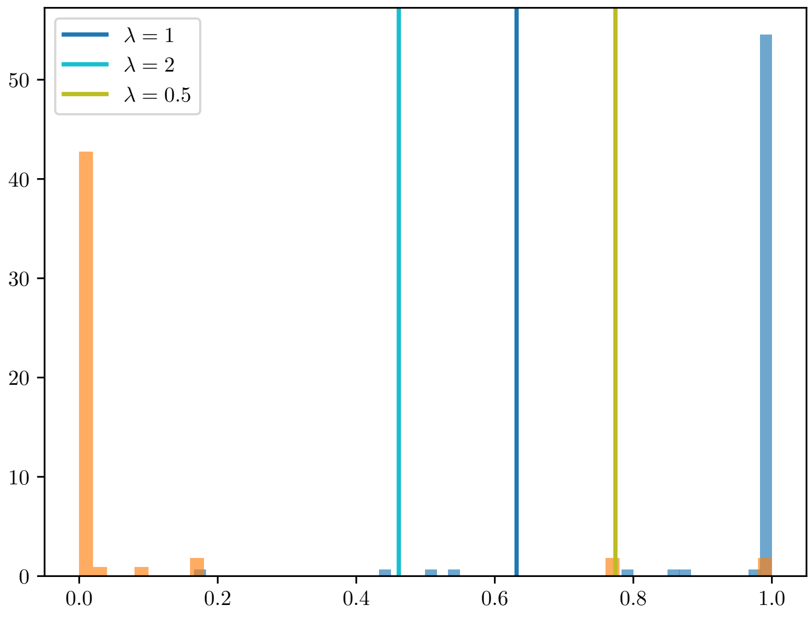

Figure 2: Thresholds and predictions generated for the Wisconsin Breast Cancer dataset, this is a very easy dataset to classify

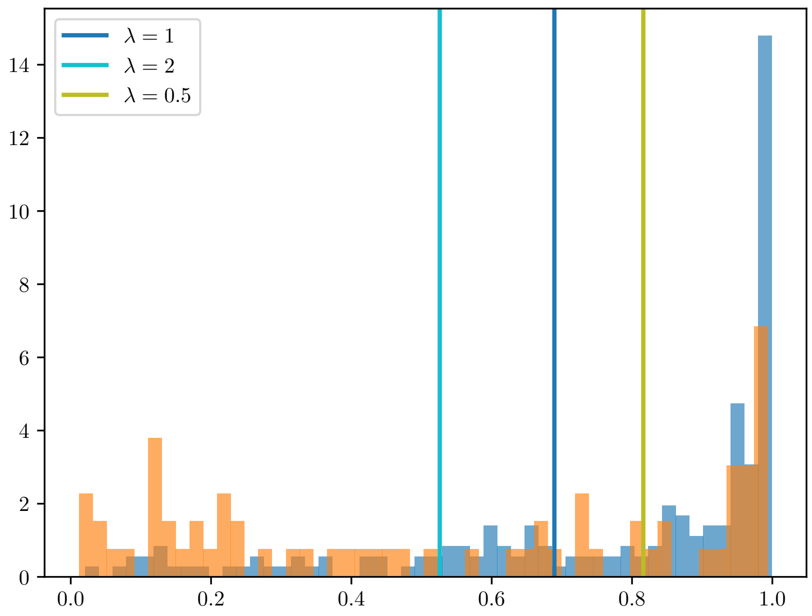

Figure 3: Thresholds and predictions generated for the German Credit Data dataset

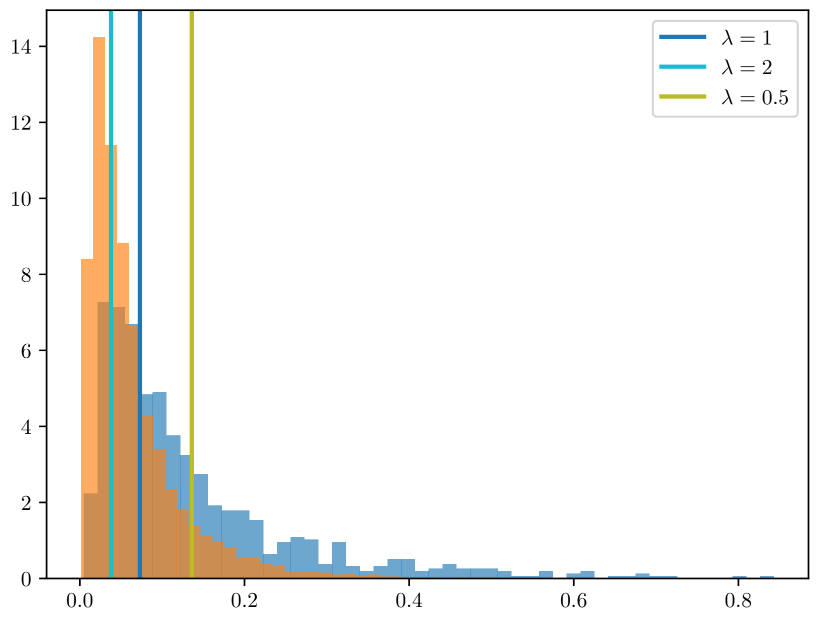

Figure 4: Thresholds and predictions generated for the KDD Cup 2009 Churn dataset