DEEP LEARNING WITH KL-DIVERGENCE: CODE DETAILS

Li Liu Di Wang Kevin He

*Corresponding author
[1]Address (first author)
[2]Address (second author)
E-mails: email@1 / email@2

ABSTRACT. Abstract should concisely summarize the key findings of the paper. It should consist of a single paragraph containing no more than 150 words. The Abstract does not have a section number.

Keywords: After the abstract three keywords must be provided.

1 Prerequisites: Discrete Time Model with KL-divergence

In this section we briefly introduce the idea in Wang et al. (2021).

1.1 Original Model

Let T_i denote the event time of interest and C_i be the censoring time for subject i with $i = 1, \ldots, n$ and n is the total sample size. The observed survival time is $X_i = \min\{T_i, C_i\}$. Let $t_1, \ldots, t_K$ be the discrete failure time indexed by $k = 1, \ldots, K$. Let $\mathcal{D}_k, \mathcal{C}_k$ be the set of labels associated with individuals failing at time t_k and censoring at time t_k, respectively. Then the likelihood is given by

$$L = \prod_{k=1}^{K} \{ \prod_{i \in \mathcal{D}_k} f(t_k; \mathbf{Z}_i) \prod_{i \in \mathcal{C}_k} S(t_k; \mathbf{Z}_i) \} \tag{1}$$

where $f(t_k; \mathbf{Z}_i) = P(T_i = t_k | \mathbf{Z}_i)$ and $S(t_k; \mathbf{Z}_i) = P(T_i > t_k | \mathbf{Z}_i) = \prod_{l:t_l \leq t_k} \{1 - \lambda(t_l; \mathbf{Z}_i)\}$ is the survival function. Denote $\lambda(t_k; \mathbf{Z}_i) = g(\gamma_k + \mathbf{Z}_i^T \boldsymbol{\beta})$ with $g(u) = h^{-1}(u)$, then the log-likelihood is given by

$$l(\boldsymbol{\theta}) = \sum_{i=1}^{n} \sum_{k=1}^{K} Y_i(t_k) \left[\delta_i(t_k) \log \left\{ \frac{g(\gamma_k + \mathbf{Z}_i^T \boldsymbol{\beta})}{1 - g(\gamma_k + \mathbf{Z}_i^T \boldsymbol{\beta})} \right\} + \log\{1 - g(\gamma_k + \mathbf{Z}_i^T \boldsymbol{\beta})\} \right] \tag{2}$$

where $\boldsymbol{\theta} = (\boldsymbol{\gamma}^T, \boldsymbol{\beta}^T)^T$, $Y_i(t_k) = I(X_i \geq t_k)$ and $\delta_i(t_k) = I(T_i = t_k)$. Note that for censored data, $\delta_i(t_k) = 0, k = 1, \ldots, n$.

1.2 Model with KL-divergence

Suppose λ_1, λ_2 are two hazard functions corresponding to two survival functions P_1, P_2, then conditional on $T_i \geq t_k$, we have the KL-divergence

$$D_{KL}(P_1, P_2; t_k, \mathbf{Z}_i) = \lambda_1(t_k; \mathbf{Z}_i) \log \left\{ \frac{\lambda_1(t_k; \mathbf{Z}_i)}{\lambda_2(t_k; \mathbf{Z}_i)} \right\} + (1 - \lambda_1(t_k; \mathbf{Z}_i)) \log \left\{ \frac{1 - \lambda_1(t_k; \mathbf{Z}_i)}{1 - \lambda_2(t_k; \mathbf{Z}_i)} \right\} \tag{3}$$

So the KL-divergence for the subject i is defined as

$$D_{KL}(P_1, P_2; \mathbf{Z}_i) = \sum_{k=1}^{K} Y_i(t_k) D_{KL}(P_1, P_2; t_k, \mathbf{Z}_i) \tag{4}$$

Note that if P_1 is denoted as the external model and P_2 is the discrete time model in the internal dataset, then denote $\tilde{\lambda}(t_k; \mathbf{Z}_i) = P_1(T_i = t_k \mid T_i \geq t_k, \mathbf{Z}_i)$, we have

$$D_{KL}(P_1, P_2; t_k, \mathbf{Z}_i) = \tilde{\lambda}(t_k; \mathbf{Z}_i) \log \left\{ \frac{\tilde{\lambda}(t_k; \mathbf{Z}_i)}{\lambda(t_k; \mathbf{Z}_i)} \right\} + (1 - \tilde{\lambda}(t_k; \mathbf{Z}_i)) \log \left\{ \frac{1 - \tilde{\lambda}(t_k; \mathbf{Z}_i)}{1 - \lambda(t_k; \mathbf{Z}_i)} \right\} \tag{5}$$

$$= -\tilde{\lambda}(t_k; \mathbf{Z}_i) \log\{\lambda(t_k; \mathbf{Z}_k)\} - (1 - \tilde{\lambda}(t_k; \mathbf{Z}_i)) \log\{1 - \lambda(t_k; \mathbf{Z}_k)\} + C \tag{6}$$

Therefore

$$D_{KL}(P_1, P_2) = \sum_{i=1}^{n} \sum_{k=1}^{K} Y_i(t_k) D_{KL}(P_1, P_2; t_k, \mathbf{Z}_i) \tag{7}$$

$$= -\sum_{i=1}^{n} \sum_{k=1}^{K} Y_i(t_k) [\tilde{\lambda}(t_k; \mathbf{Z}_i) \log \left\{ \frac{g(\gamma_k + \mathbf{Z}_i^T \boldsymbol{\beta})}{1 - g(\gamma_k + \mathbf{Z}_i^T \boldsymbol{\beta})} \right\} + \log \left\{1 - g(\gamma_k + \mathbf{Z}_i^T \boldsymbol{\beta})\right\}] + C \tag{8}$$

That is to say, for the model with penalized log-likelihood function

$$l_\eta(\boldsymbol{\theta}) = l(\boldsymbol{\theta}) - \eta D_{KL}(P_1, P_2) \tag{9}$$

Then the likelihood function can simply be changed as

$$l_\eta(\boldsymbol{\theta}) = \sum_{i=1}^{n}\sum_{k=1}^{K} Y_i(t_k)\left[\frac{\delta_i(t_k) + \eta\tilde{\lambda}(t_k;\mathbf{Z}_i)}{\eta+1}\log\left\{\frac{g(\gamma_k + \mathbf{Z}_i^T\boldsymbol{\beta})}{1-g(\gamma_k+\mathbf{Z}_i^T\boldsymbol{\beta})}\right\} + \log\{1-g(\gamma_k+\mathbf{Z}_i^T\boldsymbol{\beta})\}\right] \tag{10}$$

And for multiple external models' case, assume the penalized function is written as

$$l_\eta(\boldsymbol{\theta}) = l(\boldsymbol{\theta}) - \sum_{m=1}^{M}\eta_m D_{KL}(P_{1,m}, P_2) \tag{11}$$

$$l_{\boldsymbol{\eta}}(\boldsymbol{\theta}) = \sum_{i=1}^{n}\sum_{k=1}^{K} Y_i(t_k)\left[\frac{\delta_i(t_k) + \sum_{m=1}^{M}\eta_m\tilde{\lambda}(t_k;\mathbf{Z}_i)}{\sum_{m=1}^{M}\eta_m+1}\log\left\{\frac{g(\gamma_k + \mathbf{Z}_i^T\boldsymbol{\beta})}{1-g(\gamma_k+\mathbf{Z}_i^T\boldsymbol{\beta})}\right\} + \log\{1-g(\gamma_k+\mathbf{Z}_i^T\boldsymbol{\beta})\}\right] \tag{12}$$

where $\boldsymbol{\eta} = (\eta_1, \ldots, \eta_m)^T$.

1.3 Code Details: Schur Component

See https://zhuanlan.zhihu.com/p/78884647

2 Deep Learning with KL-divergence

2.1 How to use Deep Learning?

Recall that

$$l_\eta(\boldsymbol{\theta}) = \sum_{i=1}^{n}\sum_{k=1}^{K} Y_i(t_k)\left[\frac{\delta_i(t_k) + \eta\tilde{\lambda}(t_k;\mathbf{Z}_i)}{\eta+1}\log\left\{\frac{g(\gamma_k + \mathbf{Z}_i^T\boldsymbol{\beta})}{1-g(\gamma_k+\mathbf{Z}_i^T\boldsymbol{\beta})}\right\} + \log\{1-g(\gamma_k+\mathbf{Z}_i^T\boldsymbol{\beta})\}\right] \tag{13}$$

We can denote term $\frac{\delta_i(t_k)+\eta\tilde{\lambda}(t_k;\mathbf{Z}_i)}{\eta+1}$ as a probability, denoting as x. And note that $\lambda(t_k;\mathbf{Z}_i) = g(\gamma_k + \mathbf{Z}_i^T\boldsymbol{\beta})$, we have

$$l_\eta(\boldsymbol{\theta}) = \sum_{i=1}^{n}\sum_{k=1}^{K} Y_i(t_k)\left[x\log\{\lambda(t_k;\mathbf{Z}_i)\} + (1-x)\log\{1-\lambda(t_k;\mathbf{Z}_i)\}\right] \tag{14}$$

This is just the Binary Cross Entropy loss with Data Expansion Operation. $\lambda(\cdot)$ is what deep learning will output.

Code (Original) Kvamme and Borgan (2019)

```
def nll_logistic_hazard(phi: Tensor, idx_durations: Tensor, events: Tensor,
                        reduction: str = 'mean') -> Tensor:
    if phi.shape[1] <= idx_durations.max():
        raise ValueError(f"Network output `phi` is too small for `idx_durations`."+
                         f" Need at least `phi.shape[1] = {idx_durations.max().item()+1}`,"+
                         f" but got `phi.shape[1] = {phi.shape[1]}`")
```

```
    if events.dtype is torch.bool:
        events = events.float()
    events = events.view(-1, 1)
    idx_durations = idx_durations.view(-1, 1)
    y_bce = torch.zeros_like(phi).scatter(1, idx_durations, events) # TODO: Data Expansion!
    bce = F.binary_cross_entropy_with_logits(phi, y_bce, reduction='none')
    loss = bce.cumsum(1).gather(1, idx_durations).view(-1)
    return _reduction(loss, reduction)
```

Code (Our Model)

```
def newly_defined_loss(phi: Tensor, combined_info: Tensor, idx_durations: Tensor, events: Tensor,
                       reduction: str = 'mean') -> Tensor:

    # if phi.shape[1] <= idx_durations.max():
    #     raise ValueError(f"Network output `phi` is too small for `idx_durations`." +
    #                      f" Need at least `phi.shape[1] = {idx_durations.max().item() + 1}`," +
    #                      f" but got `phi.shape[1] = {phi.shape[1]}`")
    if events.dtype is torch.bool:
        events = events.float()
    events = events.view(-1, 1)
    idx_durations = idx_durations.view(-1, 1)
    # y_bce = torch.zeros_like(phi).scatter(1, idx_durations, events)  # TODO: Data Expansion!
    bce = F.binary_cross_entropy_with_logits(phi, combined_info, reduction='none')
    loss = bce.cumsum(1).gather(1, idx_durations).view(-1)
    return _reduction(loss, reduction)
```

References

Kvamme, H. and Borgan, Ø. (2019). Continuous and discrete-time survival prediction with neural networks. *arXiv preprint arXiv:1910.06724*.

Wang, D., Ye, W., and He, K. (2021). Kullback-leibler-based discrete relative risk models for integration of published prediction models with new dataset. In *Survival Prediction-Algorithms, Challenges and Applications*, pages 232–239. PMLR.