2.1. Regularization Path Algorithm

Our goal is to generate the solution path for a range of hyperparameter values by repeatedly calculating the next optimal solution based on the current one. Based on the updating formula in equation (16), we can easily derive the path following algorithm with respect to the regularization parameter λ. Replacing μ with λ, we have

$$\boldsymbol{\beta}^a_{\mathcal{E}}(\lambda+\epsilon) = \boldsymbol{\beta}^a_{\mathcal{E}}(\lambda) + \epsilon \begin{bmatrix} 0 & \mathbf{y}_{\mathcal{E}}^T \\ \mathbf{y}_{\mathcal{E}} & \mathbf{K}_\gamma \end{bmatrix}^{-1} \begin{pmatrix} 0 \\ \mathbf{1} \end{pmatrix}, \quad (17)$$

where the γ value is fixed. Note that equation (17) for the solution updating is equivalent to the formula in Hastie et al. (2004). Substituting (17) into (7), we can calculate the next breakpoint at which the conditions (9)–(11) will not hold if λ is changed further. As a result, the regularization path can be explored.

3. Kernel Path Algorithm

Replacing μ by γ in equation (16), the kernel path algorithm can be derived in a similar manner. We consider the period between the lth event (with $\gamma = \gamma^l$) and the $(l+1)$th event (with $\gamma = \gamma^{l+1}$). The sets $\mathcal{E}$, $\mathcal{L}$ and $\mathcal{R}$ remain unchanged during this period. Thus we trace the solution path of $(\boldsymbol{\beta}_{\mathcal{E}}(\gamma), \beta_0(\gamma))$ as γ changes.

Theorem 1 *Suppose the optimal solution is $(\boldsymbol{\beta}^l, \beta^l_0)$ when $\gamma = \gamma^l$. Then for any γ in $\gamma^{l+1} < \gamma < \gamma^l$, we have the following results:*

- *For the points $i \in \mathcal{L} \cup \mathcal{R}$, $\beta_i = \beta^l_i$ is fixed at 0 or 1, which is independent of γ;*
- *The solution to $(\boldsymbol{\beta}_{\mathcal{E}}, \beta_0)$ is given by*

$$\begin{pmatrix} \beta_0 \\ \boldsymbol{\beta}_{\mathcal{E}} \end{pmatrix} = \begin{pmatrix} \beta^l_0 \\ \boldsymbol{\beta}^l_{\mathcal{E}} \end{pmatrix} + \begin{bmatrix} 0 & \mathbf{y}_{\mathcal{E}}^T \\ \mathbf{y}_{\mathcal{E}} & \mathbf{K}_\gamma \end{bmatrix}^{-1} \begin{pmatrix} 0 \\ \mathbf{b}_\gamma \end{pmatrix}, \quad (18)$$

where

$$\mathbf{K}_\gamma = \left[y_{\mathcal{E}(i)} y_{\mathcal{E}(j)} K_\gamma(\mathbf{x}_{\mathcal{E}(i)}, \mathbf{x}_{\mathcal{E}(j)}) \right]^m_{i,j=1}, \quad (19)$$

$$\mathbf{b}_\gamma = \left(-y_{\mathcal{E}(i)} \left[\sum_{j=1}^n \beta^l_j y_j K_\gamma(\mathbf{x}_{\mathcal{E}(i)}, \mathbf{x}_j) + \beta^l_0 \right] + \lambda \right)^m_{i=1} \quad (20)$$

As such, we can update the solution to $(\boldsymbol{\beta}_{\mathcal{E}}, \beta_0)$ exactly while those to others remain unchanged. The value of γ can either increase or decrease. As γ changes, the algorithm monitors the occurrence of any of the following events:

- One of the $\beta_{\mathcal{E}(i)}$ for $i = 1, \ldots, m$ reaches 0 or 1.
- A point $i \notin \mathcal{E}$ hits the elbow, i.e., $y_i f(\mathbf{x}_i) = 1$.

By monitoring the occurrence of these events, we compute the largest $\gamma < \gamma^l$ for which an event occurs. This γ value is a breakpoint and is denoted by γ^{l+1}. We then update the point sets and continue until the algorithm terminates.

Table 1. Kernel path algorithm.

```
Input:
  $\hat{\boldsymbol{\beta}}$, $\hat{\beta}_0$, $\gamma_0$ - initial solution for $\gamma$;
  $\theta$, $\epsilon$, $\gamma_{min}$ - decay rate, error tolerance, $\gamma$ limit

1. $t = 0$; $\boldsymbol{\beta}^0 = \hat{\boldsymbol{\beta}}, \beta^0_0 = \hat{\beta}_0$
2. while $\gamma > \gamma_{min}$
3.     $r = \theta$;
4.     while $r < 1 - \epsilon$
5.         $\gamma = r\gamma_t$;
6.         use (18) to compute $(\boldsymbol{\beta}(\gamma), \beta_0(\gamma))$
7.         if $(\boldsymbol{\beta}(\gamma), \beta_0(\gamma))$ is the valid solution
8.             $\boldsymbol{\beta}^{t+1} = \boldsymbol{\beta}(\gamma)$; $\beta^{t+1}_0 = \beta_0(\gamma)$; $\gamma_{t+1} = \gamma$;
               $t = t + 1$
9.         else $r = r^{1/2}$
10.        endif
11.    end-while
12.    update the point sets $\mathcal{E}, \mathcal{L}, \mathcal{R}$;
13.end-while ;

Output:
a sequence of solutions $(\boldsymbol{\beta}(\gamma), \beta_0(\gamma)), \gamma_{min} < \gamma < \gamma_0$
```

In the previous works (Zhu et al., 2003; Hastie et al., 2004; Wang et al., 2006), the solution path is piecewise linear with respect to some hyperparameter. The breakpoint at which the next event occurs can be calculated in advance before actually reaching it. However, the value of the kernel hyperparameter is implicitly embedded into the pairwise distance between points. As a result, we need to specify a γ value in advance to compute the next solution and then check whether the next event has occurred or not. Suppose we are given the optimal solution at $\gamma = \gamma^l$. We propose here an efficient algorithm for estimating the next breakpoint, i.e., γ^{l+1}, at which the next event occurs. Table 1 shows the pseudocode of our proposed kernel path algorithm. The user has to specify a decay rate $\theta \in (0, 1)$. At each iteration, γ is decreased through multiplying it by θ. If the next event has not occurred, we continue to multiply γ by θ. Otherwise the decay rate is set to $\theta^{1/2}$. The above steps are repeated until the decay rate becomes less than $(1 - \epsilon)$, where ϵ is some error tolerance specified in advance by the user. Hence, we can estimate the breakpoint γ such that