Supplementary Material: Fast Sparse Classification for Generalized Linear and Additive Models

A THEOREMS AND PROOFS

A.1 Thresholding Is Too Conservative

The first theorem shows that thresholding is too conservative. Recall that with the support set fixed (i.e., $\lambda_0 = 0$), the loss can be written as $G(\boldsymbol{w}) = \sum_{i=1}^n \log(1 + \exp(-y_i(\boldsymbol{x}_i^T \boldsymbol{w}))) + \lambda_2 \|\boldsymbol{w}\|_2^2$.

Theorem 4.1 *(Thresholding is too conservative.) Let $\boldsymbol{w}^t$ be the current solution at iteration t, w_j^t be the coefficient for the j-th feature, and let w_j^* be the optimal value on the j-th coefficient while keeping all other coefficients fixed to their values at time t. Furthermore, let $\boldsymbol{w}^{t+1} = \boldsymbol{w}^t + \boldsymbol{e}_j(T(j, \boldsymbol{w}^t) - w_j^t)$, where $\boldsymbol{e}_j$ is a vector with 1 on the j-th component and 0 otherwise and $T(j, \boldsymbol{w}^t)$ is the thresholding operation with the support set fixed (i.e., $\lambda_0 = 0$). Then we have the following inequalities:*

$$\begin{aligned}\nabla_j G(\boldsymbol{w}^t)\nabla_j G(\boldsymbol{w}^{t+1}) &\geq 0,\\ (w_j^t - w_j^*)(w_j^{t+1} - w_j^*) &\geq 0,\\ \text{and } G(\boldsymbol{w}^t) &\geq G(\boldsymbol{w}^{t+1}).\end{aligned}$$

Proof.
For notational convenience, let us define two functions:

$$\begin{aligned}F(u) &:= G(\boldsymbol{w}^t) + (u - w_j^t)\nabla_j G(\boldsymbol{w}^t) + \frac{1}{2}L_j(u - w_j^t)^2\\ H(u) &:= G(\boldsymbol{w}^t - w_j^t\boldsymbol{e}_j + u\boldsymbol{e}_j).\end{aligned}$$

Using the notation above, our thresholding operation can be rewritten as $T(j, \boldsymbol{w}) \in \arg\min_u F(u)$. This means $w_j^{t+1} = T(j, \boldsymbol{w})$ minimizes $F(\cdot)$. After the thresholding operation, we update $\boldsymbol{w}$ by $\boldsymbol{w}^{t+1} = \boldsymbol{w}^t - w_j^t\boldsymbol{e}_j + w_j^{t+1}\boldsymbol{e}_j$. Furthermore, we use w_j^* to denote the optimal value that minimizes $H(\cdot)$. Throughout this proof, we assume $\lambda_0 = 0$ because the support set is fixed.

Using the new notation for $F(u)$ and $H(u)$, we have the following expression for their first and second derivatives:

$$\begin{aligned}F'(u) &= \nabla_j G(\boldsymbol{w}^t) + L_j(u - w_j^t), & F''(u) &= L_j\\ H'(u) &= \nabla_j G(\boldsymbol{w}^t - w_j^t\boldsymbol{e}_j + u\boldsymbol{e}_j), & H''(u) &= \nabla_{jj}^2 G(\boldsymbol{w}^t - w_j^t\boldsymbol{e}_j + u\boldsymbol{e}_j)\\ F'(w_j^t) &= H'(w_j^t), & 0 \leq H''(u) &\leq L_j = F''(u).\end{aligned}$$

To get $F'(w_j^t) = H'(w_j^t)$, we plug in $u = w_j^t$ into the formula for $F'(u)$ and $H'(u)$. For the last inequalities, we have $H''(u) \geq 0$ because $H(u)$ is a convex function. In addition, we have $H''(u) \leq L_j$ because L_j is the Lipschitz constant for $H'(u)$ so that $|H'(u+d) - H'(u)| \leq L_j|d|$ and $|H''(u)| = \lim_{d\to 0} |\frac{H'(u+d)-H'(u)}{d}| \leq L_j$.

Note that $F(u)$ is a quadratic upper bound of $H(u)$. First we have that $F - H$ is a convex function because the second derivative of $F - H$ is greater than or equal to 0. Second, the first derivative of $F - H$ at w_j^t is 0. Third, $F - H$ at w_j^t is also 0. These three things mean that $F(u) - H(u) \geq 0$ for any $u \in \mathbb{R}$. Therefore, $F(u)$ is a quadratic upper bound of $H(u)$.

We want to show

$$\begin{aligned}\nabla_j G(\boldsymbol{w}^t)\nabla_j G(\boldsymbol{w}^{t+1}) &\geq 0,\\ (w_j^t - w_j^*)(w_j^{t+1} - w_j^*) &\geq 0,\\ \text{and } G(\boldsymbol{w}^t) &\geq G(\boldsymbol{w}^{t+1}).\end{aligned}$$

Using the new notation, it is equivalent for us to show

$$H'(w_j^t)H'(w_j^{t+1}) \geq 0, \tag{11}$$

$$(w_j^t - w_j^*)(w_j^{t+1} - w_j^*) \geq 0, \tag{12}$$

$$\text{and } H(w_j^t) \geq H(w_j^{t+1}). \tag{13}$$

To show the inequalities above, we discuss three cases: Case 1) $w_j^t < w_j^*$, Case 2) $w_j^t > w_j^*$, and Case 3) $w_j^t = w_j^*$.

Case 1: $w_j^t < w_j^*$

If $w_j^t < w_j^*$, we have $H'(w_j^t) < 0$. This is true because

$$\begin{aligned} H'(w_j^t) &= H'(w_j^*) + \int_{w_j^*}^{w_j^t} H''(u)du \\ &= 0 + \int_{w_j^*}^{w_j^t} H''(u)du \\ &= -\int_{w_j^t}^{w_j^*} H''(u)du < 0. \end{aligned}$$

The last inequality holds because $H''(u) \geq 0$ and $H''(u) > 0$ for some nonzero measurable set in $[w_j^t, w_j^*]$.

Now because w_j^{t+1} minimizes $F(\cdot)$, we have $F'(w_j^{t+1}) = 0$. Using the relationship between $F(\cdot)$ and $H(\cdot)$, we have

$$\begin{aligned} 0 = F'(w_j^{t+1}) &= F'(w_j^t) + \int_{w_j^t}^{w_j^{t+1}} F''(u)du \\ &= H'(w_j^t) + \int_{w_j^t}^{w_j^{t+1}} F''(u)du \\ &\geq H'(w_j^t) + \int_{w_j^t}^{w_j^{t+1}} H''(u)du \\ &= H'(w_j^{t+1}). \end{aligned}$$

Therefore, we have $H'(w_j^{t+1}) \leq 0$. Since $H'(w_j^t) < 0$, we have $H'(w_j^t)H'(w_j^{t+1}) \geq 0$, proving (11) for Case 1.

Let us prove (12) for Case 1. For the sake of contradiction, suppose $w_j^{t+1} > w_j^*$, we have $H'(w_j^{t+1}) > 0$ because

$$\begin{aligned} H'(w_j^{t+1}) &= H'(w_j^*) + \int_{w_j^*}^{w_j^{t+1}} H''(u)du \\ &= 0 + \int_{w_j^*}^{w_j^{t+1}} H''(u)du \\ &= \int_{w_j^*}^{w_j^{t+1}} H''(u)du > 0. \end{aligned}$$

This implies $H'(w_j^t)H'(w_j^{t+1}) < 0$, contradicting the proof of (11) for Case 1 above. Thus, we have $w_j^{t+1} \leq w_j^*$ and $(w_j^t - w_j^*)(w_j^{t+1} - w_j^*) \geq 0$, proving (12) for Case 1.

Lastly, because of the relationship between $F(\cdot)$ and $H(\cdot)$, we have

$$H(w_j^{t+1}) \leq F(w_j^{t+1}) = \min_u F(u) \leq F(w_j^t) = H(w_j^t).$$

This proves our third inequality (13) for Case 1.

Case 2: $w_j^t > w_j^*$

If $w_j^t > w_j^*$, we have $H'(w_j^t) > 0$. The procedure to show this is very similar to what we have shown in Case 1, so we omit it here.

Now because w_j^{t+1} minimizes $F(\cdot)$, we have $F'(w_j^{t+1}) = 0$. Using the relationship between $F(\cdot)$ and $H(\cdot)$, we have

$$\begin{aligned} 0 = F'(w_j^{t+1}) &= F'(w_j^t) + \int_{w_j^t}^{w_j^{t+1}} F''(u)du \\ &= H'(w_j^t) + \int_{w_j^t}^{w_j^{t+1}} F''(u)du \\ &\leq H'(w_j^t) + \int_{w_j^t}^{w_j^{t+1}} H''(u)du \\ &= H'(w_j^{t+1}). \end{aligned}$$

The third line holds true because $w_j^{t+1} < w_j^t$. Therefore, we have $H'(w_j^{t+1}) \geq 0$. Since $H'(w_j^t) > 0$, we have $H'(w_j^t)H'(w_j^{t+1}) \geq 0$, proving the inequality (11) under Case 2.

We proceed to prove (12) for Case 2. Now for the sake of contradiction, suppose $w_j^{t+1} < w_j^*$, we have $H'(w_j^{t+1}) < 0$. Again, the procedure to show this is very similar to what we have shown in Case 1, so we omit it here. This reasoning implies $H'(w_j^t)H'(w_j^{t+1}) < 0$, contradicting the proof of (11) under Case 2 above. Thus, we have $w_j^{t+1} \geq w_j^*$ and $(w_j^t - w_j^*)(w_j^{t+1} - w_j^*) \geq 0$, proving (12) for Case 2.

The procedure to prove (13) under Case 2 identical to (13) under Case 1, so we omit the proof here.

Case 3: $w_j^t = w_j^*$.

In this special case, $w_j^{t+1} = w_j^t$. Thus, the three inequalities (11), (12), and (13) hold trivially. This completes the proof for Theorem 4.1.

To the best of our knowledge, we are the first to show that w_j^{t+1} and w_j^t stay on the same side of w_j^*. This important point is the motivation behind our use of cutting planes and the development of our quadratic lower bound.

A.2 Lower Bound via Cutting Planes

Theorem 4.2 *Suppose $f(x)$ is convex and differentiable on domain $\mathbb{R}$. Let α_1 and α_2 be slopes of tangent lines of $f(x)$ at locations x_1 and x_2. If $\alpha_1\alpha_2 \leq 0$, there is a lower bound on the optimal value $f(x^*)$:*

$$f(x^*) \geq \frac{\alpha_1 f(x_2) - \alpha_2 f(x_1) + \alpha_1\alpha_2(x_1 - x_2)}{\alpha_1 - \alpha_2}.$$

Proof.
Because of the convexity of $f(x)$, we have

$$\begin{aligned} f(x) &\geq f(x_1) + \alpha_1(x - x_1) \quad \text{where } \alpha_1 = f'(x_1) \\ f(x) &\geq f(x_2) + \alpha_2(x - x_2) \quad \text{where } \alpha_2 = f'(x_2). \end{aligned}$$

Notice that the function $f(x)$ sits above two lines $y = f(x_1) + \alpha_1(x - x_1)$ and $y = f(x_2) + \alpha_2(x - x_2)$.

Equating these two lines to find the intersection point $\hat{x}$, we have

$$\begin{aligned} & f(x_1) + \alpha_1(\hat{x} - x_1) = f(x_2) + \alpha_2(\hat{x} - x_2) \qquad (14) \\ \Rightarrow & \hat{x} = \frac{f(x_2) - f(x_1) + \alpha_1 x_1 - \alpha_2 x_2}{\alpha_1 - \alpha_2}. \end{aligned}$$

To find the intersection value $\hat{y}$, we plug in $\hat{x}$ into the left side of (14) and get

$$\begin{aligned}\hat{y} &= f(x_1) + \alpha_1(\hat{x} - x_1)\\ &= \frac{\alpha_1 f(x_2) - \alpha_2 f(x_1) + \alpha_1\alpha_2(x_1 - x_2)}{\alpha_1 - \alpha_2}.\end{aligned}$$

Since the function $f(x)$ sits above the two lines and therefore above the intersection value $\hat{y}$, we have

$$f(x^*) \geq \frac{\alpha_1 f(x_2) - \alpha_2 f(x_1) + \alpha_1\alpha_2(x_1 - x_2)}{\alpha_1 - \alpha_2}.$$

This completes the proof for Theorem 4.2.

A.3 Lower Bound via Quadratic Cuts

Theorem 4.3 *Suppose $f(x) = g(x) + \lambda_2 x^2$, where $g(x)$ is a convex and differentiable function. Then $f(x)$ is strongly convex. Let α_1 be the slope of the tangent line to $f(x)$ at location x_1. Then, there is a lower bound on the optimal value $f(x^*)$:*

$$f(x^*) \geq \mathcal{L}_{low} := f(x_1) - \frac{\alpha_1^2}{4\lambda_2}. \tag{15}$$

Let α_2 be the slope of the tangent line to $f(x)$ at another location x_2. If $\alpha_1\alpha_2 \leq 0$, a lower bound on the optimal value $f(x^*)$ is as follows:

$$f(x^*) \geq \mathcal{L}_{low} := f(\hat{x}) + \alpha_1(\hat{x} - x_1) + \lambda_2(\hat{x} - x_1)^2, \tag{16}$$

where

$$\hat{x} = \frac{-f(x_1) + f(x_2) + \alpha_1 x_1 - \alpha_2 x_2 - \lambda_2(x_1^2 - x_2^2)}{\alpha_1 - \alpha_2 - 2\lambda_2(x_1 - x_2)}.$$

Proof.
Given a convex function $g(x)$, we first show that $f(x) = g(x) + \lambda_2 x^2$ is a strongly convex function before proving the two bounds in Theorem 4.3.

To show that $f(x)$ is a strongly convex function, it is sufficient to show

$$f(y) \geq f(x) + f'(x)(y - x) + \lambda_2(y - x)^2 \tag{17}$$

for any $x, y \in \mathbb{R}$.

Because $g(x)$ is convex, we have

$$g(y) \geq g(x) + g'(x)(y - x).$$

Adding $\lambda_2 y^2$ to both sides, we have

$$g(y) + \lambda_2 y^2 \geq g(x) + g'(x)(y - x) + \lambda_2 y^2.$$

The LHS is $f(y)$. The RHS can be rewritten as

$$\begin{aligned}&g(x) + g'(x)(y - x) + \lambda_2 y^2\\ &\quad= \quad g(x) + \lambda_2 x^2 + (g'(x) + 2\lambda_2 x)(y - x) + \lambda_2 y^2 - \lambda_2 x^2 - 2\lambda_2 x(y - x)\\ &\quad= \quad f(x) + f'(x)(y - x) + \lambda_2(x - y)^2.\end{aligned}$$

Therefore, $f(x)$ is a strongly convex function.

Because $f(\cdot)$ is strongly convex, where (17) holds for any x and y, given a point x_1 with $\alpha_1 := f'(x_1)$, then our strongly convex function $f(\cdot)$ at any point x is bounded by

$$f(x) \geq f(x_1) + \alpha_1(x - x_1) + \lambda_2(x - x_1)^2.$$

The RHS is a quadratic function of x, with the minimum value achieved at $f(x_1) - \frac{\alpha_1^2}{4\lambda_2}$, so we have

$$f(x) \geq f(x_1) - \frac{\alpha_1^2}{4\lambda_2}.$$

Since the above inequality works for any $x \in \mathbb{R}$, it also works for the optimal value x^*:

$$f(x^*) \geq f(x_1) - \frac{\alpha_1^2}{4\lambda_2}.$$

Therefore, we have proved (15).

Suppose we are given another point x_2 with $\alpha_2 := f'(x_2)$, then $f(x)$ sits above two quadratic equations:

$$\begin{aligned} f(x) &\geq f(x_1) + \alpha_1(x - x_1) + \lambda_2(x - x_1)^2 \\ f(x) &\geq f(x_2) + \alpha_2(x - x_2) + \lambda_2(x - x_2)^2. \end{aligned}$$

Equating these two quadratic equations to find the intersection point $\hat{x}$, we have

$$\begin{aligned} &f(x_1) + \alpha_1(\hat{x} - x_1) + \lambda_2(\hat{x} - x_1)^2 = f(x_2) + \alpha_2(\hat{x} - x_2) + \lambda_2(\hat{x} - x_2)^2 \\ \Rightarrow &\hat{x} = \frac{-f(x_1) + f(x_2) + \alpha_1 x_1 - \alpha_2 x_2 + \lambda_2(x_2^2 - x_1^2)}{\alpha_1 - \alpha_2 - 2\lambda_2(x_1 - x_2)}. \end{aligned}$$

Plugging in the intersection point $\hat{x}$, we can get the intersection value $\hat{y}$, which is a lower bound of $f(x^*)$

$$f(x^*) \geq \hat{y} = f(x_1) + \alpha_1(\hat{x} - x_1) + \lambda_2(\hat{x} - x_1)^2.$$

This completes the proof for (16).

A.4 Derivation for the Exponential Loss

The exponential loss function is defined as $H(\boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^{n} e^{-y_i f(\mathbf{x}_i)}$, where $f(\boldsymbol{x}_i) = \boldsymbol{w}^T \boldsymbol{x}_i$. Since $\boldsymbol{x}_i$ is a binary vector, s.t. $x_{ij} \in \{-1, 1\}$ and $y_i \in \{-1, 1\}$, let $\boldsymbol{z}_i = y_i \boldsymbol{x}_i$ and $\boldsymbol{z}_i \in \{-1, 1\}^p$. After t iterations, the exponential loss function can be written as:

$$H(\boldsymbol{w}^t) = \frac{1}{n}\sum_{i=1}^{n} e^{-y_i(\sum_{j=1}^{p} w_j^t x_{ij})} = \frac{1}{n}\sum_{i=1}^{n} e^{-(\boldsymbol{w}^t)^T \boldsymbol{z}_i}.$$

We will perform a linesearch, where we optimize coefficient j at iteration t. This linesearch optimization problem for coordinate j is $w_j^{t+1} \in \arg\min_w H([w_1^t, ..., w_{j-1}^t, w, w_{j+1}^t, ...]) + \lambda_0 \|[w_1^t, ..., w_{j-1}^t, w, w_{j+1}^t, ...]\|_0$.

Theorem 5.1 *Let $\boldsymbol{w}^t$ be the coefficient vector at iteration t, $H^t := H(\boldsymbol{w}^t)$ and λ_0 be the regularization constant for the ℓ_0 penalty. For the j-th coordinate, we update the coefficient according to:*

(1) Suppose $w_j^t = 0$. Let $d_- = \sum_{i: z_{ij} = -1} c_i / \sum_{i=1}^{n} c_i$, where $c_i = e^{-(\boldsymbol{w}^t)^T \boldsymbol{z}_i}$. Then, if d_- is within the interval:

$$\left[\frac{1}{2} - \frac{1}{2H^t}\sqrt{\lambda_0(2H^t - \lambda_0)}, \frac{1}{2} + \frac{1}{2H^t}\sqrt{\lambda_0(2H^t - \lambda_0)}\right],$$

then set w_j^{t+1} to 0. Otherwise set $w_j^{t+1} = \frac{1}{2}\ln\frac{1-d_-}{d_-}$.

(2) Suppose $w_j^t \neq 0$. Let $D_- = \sum_{i: z_{ij} = -1} c_i / \sum_{i=1}^{n} c_i$, where $c_i = e^{-(\boldsymbol{w}^t - w_j^t \boldsymbol{e}_j)^T z_i}$. Let $H_{\neg j}^t = H(\boldsymbol{w}^t - w_j^t \boldsymbol{e}_j)$. Then, if D_- is within the interval:

$$\left[\frac{1}{2} - \frac{1}{2H_{\neg j}^t}\sqrt{\lambda_0(2H_{\neg j}^t - \lambda_0)}, \frac{1}{2} + \frac{1}{2H_{\neg j}^t}\sqrt{\lambda_0(2H_{\neg j}^t - \lambda_0)}\right],$$

then set w_j^{t+1} to 0. Otherwise, set $w_j^{t+1} = \frac{1}{2}\ln\frac{1-D_-}{D_-}$.

While these expressions may first appear difficult to calculate, they are not. Like AdaBoost, we make multiplicative updates to the loss at each iteration. Thus, since H^t is easy to calculate, $H^t_{\neg j}$ is also easy to calculate (requiring only a multiplication), and the rest is simple mathematical operations.

Intuitively, using AdaBoost's terminology, the bound states that if the weak learning algorithm produces a stronger weak classifier at that iteration (a classifier whose error rate is away from 1/2), we would keep it. Otherwise, we would not; we would rather set its coefficient to 0. In some sense, this result is reminiscent of iterative thresholding (Daubechies et al., 2004).

Proof.
Case 1: Suppose at iteration t, $w_j^t = 0$ and in the next iteration $t+1$, we evaluate placing feature j into the model, i.e., set $w_j^{t+1} \neq 0$. Then, the decrease in loss should be larger than λ_0, otherwise $w_j^{t+1} = 0$. Suppose we want to add feature j into the model, the loss function is

$$H^{t+1} = \frac{1}{n}\sum_{i=1}^{n} e^{-(\boldsymbol{w}^t)^T \boldsymbol{z}_i - y_i w_j x_{ij}} = \frac{1}{n}\sum_{i=1}^{n} e^{-(\boldsymbol{w}^t)^T \boldsymbol{z}_i - w_j z_{ij}}.$$

We can get an analytical solution for w_j by solving $\frac{\partial H^{t+1}}{\partial w_j} = 0$, which is the same as AdaBoost's update step.

$$\begin{aligned} 0 = \frac{\partial H^{t+1}}{\partial w_j}\Big|_{w_j^*} &= \sum_{i=1}^{n} -z_{ij} e^{-(\boldsymbol{w}^t)^T \boldsymbol{z}_i} e^{-w_j z_{ij}}\Big|_{w_j^*} \\ &= \sum_{i: z_{ij}=1} -e^{-(\boldsymbol{w}^t)^T \boldsymbol{z}_i} e^{-w_j}\Big|_{w_j^*} + \sum_{i: z_{ij}=-1} e^{-(\boldsymbol{w}^t)^T \boldsymbol{z}_i} e^{w_j}\Big|_{w_j^*}. \end{aligned} \tag{18}$$

Multiplying by a normalization constant

$$C = \sum_{i=1}^{n} c_i = \sum_{i=1}^{n} e^{-(\boldsymbol{w}^t)^T \boldsymbol{z}_i},$$

and defining

$$d_+ = \frac{\sum_{i: z_{ij}=1} e^{-(\boldsymbol{w}^t)^T \boldsymbol{z}_i}}{C} \quad \text{and} \quad d_- = \frac{\sum_{i: z_{ij}=-1} e^{-(\boldsymbol{w}^t)^T \boldsymbol{z}_i}}{C},$$

Equation (18) becomes

$$0 = -d_+ e^{-w_j^*} + d_- e^{w_j^*}.$$

Solving this yields:

$$w_j^* = \frac{1}{2}\ln\frac{d_+}{d_-}.$$

Recalling that $d_+ = 1 - d_-$, the lowest possible loss after adding in feature j is thus:

$$H^{t+1} = \mathcal{L}^t \cdot \left((1-d_-)\left(\frac{1-d_-}{d_-}\right)^{-1/2} + d_-\left(\frac{1-d_-}{d_-}\right)^{1/2} \right) = H^t \cdot 2\left((1-d_-)d_-\right)^{1/2}. \tag{19}$$

We have now derived the best possible value for w_j if it were nonzero. However, our objective suffers a penalty of λ_0 from the regularization term whenever w_j is nonzero. Thus, we need to compare the objective with $w_j = 0$ to the regularized objective with (19) as the loss term. If the difference is less than λ_0, it would benefit the objective to set coefficient j to 0 at the next iteration. The condition for setting w_j to 0 is:

$$\begin{aligned} H^t - H^{t+1} = H^t - H^t \cdot 2\left((1-d_-)d_-\right)^{1/2} &\leq \lambda_0. \\ \frac{H^t - \lambda_0}{2H^t} &\leq ((1-d_-)d_-)^{1/2} \\ \left(\frac{H^t - \lambda_0}{2H^t}\right)^2 &\leq (1-d_-)d_- \\ d_-^2 - d_- + \left(\frac{H^t - \lambda_0}{2H^t}\right)^2 &\leq 0. \end{aligned} \tag{20}$$

This is a quadratic equation, permitting solutions in $d_- \in \left(\frac{1}{2} - \frac{1}{2H^t}\sqrt{\lambda_0(2H^t - \lambda_0)}, \frac{1}{2} + \frac{1}{2H^t}\sqrt{\lambda_0(2H^t - \lambda_0)}\right)$.

Therefore, if $d_- \in \left(\frac{1}{2} - \frac{1}{2H^t}\sqrt{\lambda_0(2H^t - \lambda_0)}, \frac{1}{2} + \frac{1}{2H^t}\sqrt{\lambda_0(2H^t - \lambda_0)}\right)$, then set w_j^{t+1} to 0. Otherwise, $w_j^{t+1} = \frac{1}{2}\ln\frac{1-d_-}{d_-}$.

Case 2: Suppose at iteration t, $w_j^t \neq 0$, and in the next iteration $t+1$, we evaluate updating w_j^t. Then the decrease in loss should be larger than $H^t - H^t_{\neg j} + \lambda_0$, otherwise, $w_j^{t+1=0}$. Suppose we want to update w_j at iteration $t+1$, the loss function is

$$H^{t+1} = \frac{1}{n}\sum_{i=1}^{n} e^{-(\boldsymbol{w}^t - w_j^t \boldsymbol{e}_j)^T \boldsymbol{z}_i - w_j z_{ij}} \quad .$$

Similar to the derivation for Case 1, we can get an analytical solution for w_j by solving $\frac{\partial H^{t+1}}{\partial w_j} = 0$.

$$\begin{aligned} 0 = \frac{\partial H^{t+1}}{\partial w_j}\Big|_{w_j^*} &= \sum_{i=1}^{n} -z_{ij} e^{-(\boldsymbol{w}^t - w_j^t \boldsymbol{e}_j)^T \boldsymbol{z}_i} e^{(-w_j z_{ij})}\Big|_{w_j^*} \\ &= \sum_{i:z_{ij}=1} -e^{-(\boldsymbol{w}^t - w_j^t \boldsymbol{e}_j)^T \boldsymbol{z}_i} e^{-w_j}\Big|_{w_j^*} + \\ &\quad \sum_{i:z_{ij}=-1} e^{-(\boldsymbol{w}^t - w_j^t \boldsymbol{e}_j)^T \boldsymbol{z}_i} e^{w_j}\Big|_{w_j^*} \quad . \end{aligned} \tag{21}$$

Similarly, multiplying by a normalization constant C, and defining $D_+ = \sum_{i:z_{ij}=1} e^{-(\boldsymbol{w}^t - w_j^t \boldsymbol{e}_j)^T \boldsymbol{z}_i}/C$ and $D_- = \sum_{i:z_{ij}=-1} e^{-(\boldsymbol{w}^t - w_j^t \boldsymbol{e}_j)^T \boldsymbol{z}_i}/C$.

Then Equation 21 becomes

$$0 = -D_+ e^{-w_j^*} + D_- e^{w_j^*}.$$

Solving this yields:

$$w_j^* = \frac{1}{2}\ln\frac{D_+}{D_-}.$$

The lowest possible loss after updating the coefficient of feature j is

$$H^{t+1} = H^t_{\neg j} \cdot \left(D_+\left(\frac{D_+}{D_-}\right)^{-1/2} + D_-\left(\frac{D_+}{D_-}\right)^{1/2}\right) = H^t_{\neg j} \cdot 2((1-D_-)D_-)^{1/2}. \tag{22}$$

Similarly to Case 1, we need to compare the objective with $w_j = 0$ to the regularized objective with (22) as the loss term. If the difference is less than λ_0, it would benefit the objective to set coefficient j to 0 at the next iteration. The condition for setting w_j to 0 is:

$$H^t_{\neg j} - H^{t+1} = H^t_{\neg j} - H^t_{\neg j} \cdot 2\left((1-D_-)D_-\right)^{1/2} \leq \lambda_0.$$

Using the same derivation as in Equation (20), the solution is in

$$D_- \in \left(\frac{1}{2} - \frac{1}{2H^t_{\neg j}}\sqrt{\lambda_0(2H^t_{\neg j} - \lambda_0)}, \frac{1}{2} + \frac{1}{2H^t_{\neg j}}\sqrt{\lambda_0(2H^t_{\neg j} - \lambda_0)}\right).$$

Therefore, if $D_- \in \left(\frac{1}{2} - \frac{1}{2H^t_{\neg j}}\sqrt{\lambda_0(2H^t_{\neg j} - \lambda_0)}, \frac{1}{2} + \frac{1}{2H^t_{\neg j}}\sqrt{\lambda_0(2H^t_{\neg j} - \lambda_0)}\right)$, then set w_j^{t+1} to 0. Otherwise, $w_j^{t+1} = \frac{1}{2}\ln\frac{1-D_-}{D_-}$.

B PSEUDOCODE

We begin with the presentation of our high-level Algorithm 1 and then elaborate on the novel steps in the following lower-level algorithms.

Shortly, we discuss how TryDeleteOrSwap($\boldsymbol{w}, j, S^c$) is implemented in detail. After that, we discuss its subroutine algorithms TryAddLinCut($\boldsymbol{w}', j', \mathcal{L}_{best}$) and TryAddQuad($\boldsymbol{w}', j', \mathcal{L}_{best}$), as well as their subroutine algorithm FindNewCoefficient($\boldsymbol{w}', j'$). For algorithms TryAddLinCut($\boldsymbol{w}', j', \mathcal{L}_{best}$) and TryAddQuad($\boldsymbol{w}', j', \mathcal{L}_{best}$), we use $f(x) = G(\boldsymbol{w}' + \boldsymbol{e}_{j'} x)$ for notational convenience (assuming $w'_{j'} = 0$; if it is not, notation can be adjusted appropriately). Also for notational convenience, we use one lower bound from classical cutting planes and two lower bounds from quadratic cuts:

$$\begin{aligned}
\text{LinCut}(a, b, f(\cdot)) &= \frac{f'(a)f(b) - f'(b)f(a) + f'(a)f'(b)(a-b)}{f'(a) - f'(b)} \\
\text{QuadCut1}(a, f(\cdot)) &= f(a) - \frac{f'(a)^2}{4\lambda_2} \\
\text{QuadCut2}(a, b, f(\cdot)) &= f(a) + f'(a)(\hat{x} - a) + \lambda_2(\hat{x} - a)^2 \\
\text{with } \hat{x} &= \frac{-f(a) + f(b) + f'(a)a - f'(b)b + \lambda_2(b^2 - a^2)}{f'(a) - f'(b) - 2\lambda_2(a-b)}.
\end{aligned}$$

Algorithm 1 General Algorithm for Swapping Features

Input: coefficients $\boldsymbol{w}$ from a warm start algorithm, $\boldsymbol{c} = \boldsymbol{0}$ is a vector of size p where each c_j for $j < p$ indicates the number of times we failed to find a feature to swap with j.

Output: updated coefficients $\boldsymbol{w}$ that is a swap 1-OPT solution.

```
1: while True do
2:     Update support $S = \{j | w_j \neq 0\}$.
3:     $\Pi(S)$ = Sort($S$) according to $c_j$ for $j \in S$.     #Sort support in ascending order of the no. of failed swaps
4:     for $j$ in $\Pi(S)$ do
5:         $\boldsymbol{w}'$ = TryDeleteOrSwap($\boldsymbol{w}, j, S^c$)     #$S^c$ is the complement of $S$
6:         if $\boldsymbol{w}' \neq \boldsymbol{w}$ then
7:             Let $\boldsymbol{w} = \boldsymbol{w}'$.     #Swap was successful
8:             Go to line 2.
9:         else
10:            $c_j = c_j + 1$.     #No better feature can replace feature j
11:        end if
12:    end for
13:    Return $\boldsymbol{w}$.     #No single feature can be replaced with better features
14: end while
```

Algorithm 2 TryDeleteOrSwap($\boldsymbol{w}, j, S^c$)

Input: coefficients $\boldsymbol{w}$, feature index j with $w_j \neq 0$, set of feature indices $S^c = \{j' | w_{j'} = 0\}$.
Output: updated coefficients $\boldsymbol{w}'$ with feature j possibly deleted or swapped with feature $j' \in S^c$.

```
1: Calculate the best current loss $\mathcal{L}_{best} = G(\boldsymbol{w})$.
2: Let $\boldsymbol{w}' = \boldsymbol{w}$ and then set $w'_j = 0$.    #Drop feature j from the support
3: if $G(\boldsymbol{w}') \leq \mathcal{L}_{best}$ then
4:     Update $\boldsymbol{w}'$ with support restricted to $S \setminus \{j\}$.    #Dropping feature j leads to smaller loss
5:     Return $\boldsymbol{w}'$.
6: end if
7: Calculate $|\nabla_{S^c} G(\boldsymbol{w}^t)|$ on $S^c$.
8: $\Pi'$ = feature indices in $S^c$ sorted in descending order of $|\nabla_{S^c} G(\boldsymbol{w}')|$.    #Order features to possibly add in
9: for $j' \in \Pi'$ do
10:     Let $\boldsymbol{w}' = \text{TryAddQuad}(\boldsymbol{w}', j', \mathcal{L}_{best})$ if $(\lambda_2 > 0)$.    #or $\boldsymbol{w}' = \text{TryAddLinCut}(\boldsymbol{w}', j', \mathcal{L}_{best})$ if $(\lambda_2 = 0)$
11:     if $\boldsymbol{w}'$ has changed then
12:         Update full vector $\boldsymbol{w}'$ with support restricted to $S \cup \{j'\} \setminus \{j\}$.    #Swapping j with j' decreases loss
13:         Return $\boldsymbol{w}'$.
14:     end if
15: end for
16: Return $\boldsymbol{w}$.    #Since there were no better features, return original $\boldsymbol{w}$
```

Algorithm 3 TryAddLinCut($\boldsymbol{w}', j', \mathcal{L}_{best}$)

Input: coefficients $\boldsymbol{w}'$, feature index j', and current best loss $\mathcal{L}_{best}$.
Output: updated coefficients $\boldsymbol{w}'$.

```
1: Let $a = T(j', \boldsymbol{w}')$, $b = 2T(j', \boldsymbol{w}')$    #Take 2X distance suggested by thresholding operation Eq (4)
2: if $f'(0)f'(b) < 0$ then
3:     Let $c = (a + b)/2$    #Binary search
4:     if $f'(0)f'(c) < 0$ then
5:         Let $b = c$
6:     else
7:         Let $a = c$
8:     end if    #a and b are on opposite sides of $w^*_{j'}$
9:     Get $\mathcal{L}_{low} = \text{LinCut}(a, b, f(\cdot))$
10:     if $\mathcal{L}_{low} \geq \mathcal{L}_{best}$ then
11:         Return $\boldsymbol{w}'$    #Stop considering feature j' and exit early
12:     end if
13:     Let $\hat{w}_{j'} = \text{FindNewCoefficient}(\boldsymbol{w}', j')$.
14:     if $f(\hat{w}_{j'}) < \mathcal{L}_{best}$ then
15:         Return $\boldsymbol{w}'$ with $w'_{j'} = \hat{w}_{j'}$    #Swap feature j with feature j'
16:     end if
17:     Return $\boldsymbol{w}'$    #Eliminate considering feature j'
18: end if
19: Let $a = 2T(j', \boldsymbol{w}')$, $b = 3T(j', \boldsymbol{w}')$    #Take 3X distance suggested by thresholding operation
20: if $f'(0)f'(b) < 0$ then
21:     Go to line 9.
22: else
23:     Go to line 13.    #Minimum is far from starting point. Swap the feature to see if there's improvement.
24: end if
```

Algorithm 4 TryAddQuad($\boldsymbol{w}', j', \mathcal{L}_{best}$)

Input: coefficients $\boldsymbol{w}'$, feature index j', and current best loss $\mathcal{L}_{best}$
Output: updated coefficients $\boldsymbol{w}'$

```
1: Get $\mathcal{L}_{low} = \text{QuadCut1}(0, f(\cdot))$
2: if $\mathcal{L}_{low} \geq \mathcal{L}_{best}$ then
3:     Return $\boldsymbol{w}'$                                  #Stop considering feature $j'$ and exit early
4: end if
5: Let $a = T(j', \boldsymbol{w}')$, $b = 2T(j', \boldsymbol{w}')$     #Take 2X distance suggested by thresholding operation Eq (4)
6: if $f'(0)f'(b) < 0$ then
7:     Let $c = (a + b)/2$                                      #Binary search
8:     Get $\mathcal{L}_{low} = \text{QuadCut1}(c, f(\cdot))$
9:     if $\mathcal{L}_{low} \geq \mathcal{L}_{best}$ then
10:        Return $\boldsymbol{w}'$                              #Stop considering feature $j'$ and exit early
11:    end if
12:    if $f'(0)f'(c) < 0$ then
13:        Let $b = c$
14:    else
15:        Let $a = c$
16:    end if                                                   #a and b are on opposite sides of $w^*_{j'}$
17:    Get $\mathcal{L}_{low} = \text{QuadCut2}(a, b, f(\cdot))$
18:    if $\mathcal{L}_{low} \geq \mathcal{L}_{best}$ then
19:        Return $\boldsymbol{w}'$                              #Stop considering feature $j'$ and exit early
20:    end if
21:    Let $\hat{w}_{j'} = \text{FindNewCoefficient}(\boldsymbol{w}', j')$.
22:    if $f(\hat{w}_{j'}) < \mathcal{L}_{best}$ then
23:        Return $\boldsymbol{w}'$ with $w'_{j'} = \hat{w}_{j'}$          #Swap feature $j$ with feature $j'$
24:    end if
25:    Return $\boldsymbol{w}'$                                  #Eliminate considering feature $j'$
26: end if
27: Let $a = 2T(j', \boldsymbol{w}')$, $b = 3T(j', \boldsymbol{w}')$     #Take 3X distance suggested by thresholding operation
28: Get $\mathcal{L}_{low} = \text{QuadCut1}(a, f(\cdot))$
29: if $\mathcal{L}_{low} \geq \mathcal{L}_{best}$ then
30:     Return $\boldsymbol{w}'$                                 #Stop considering feature $j'$ and exit early
31: end if
32: if $f'(0)f'(b) < 0$ then
33:     Go to line 17.
34: else
35:     Get $\mathcal{L}_{low} = \text{QuadCut1}(b, f(\cdot))$
36:     if $\mathcal{L}_{low} \geq \mathcal{L}_{best}$ then
37:         Return $\boldsymbol{w}'$                             #Stop considering feature $j'$ and exit early
38:     end if
39:     Go to line 21.    #Minimum is far from starting point. Swap the feature to see if there's improvement.
40: end if
```

Algorithm 5 FindNewCoefficient($\boldsymbol{w}', j'$)

Input: coefficients $\boldsymbol{w}$, coordinate j', iteration steps max_iter=10 (default)
Output: updated coefficient $w_{j'}$ for coordinate j'

```
1: for t in 1, 2, ..., max_iter do
2:     $\boldsymbol{w}' = \boldsymbol{w}' - w_{j'}\boldsymbol{e}_{j'} + T(\boldsymbol{w}', j')\boldsymbol{e}_{j'}$     #Apply the thresholding operation on coordinate $j'$
3: end for
4: Return $w'_{j'}$
```

C EXPERIMENTAL DETAILS

We next present the datasets used in our experiments, our preprocessing steps, and the experimental setup.

C.1 Datasets

We present results using 5 datasets: two synthetic datasets (one in which the features are highly correlated for binary classification and the other in which the features are highly correlated for linear regression), the Fair Isaac (FICO) credit risk dataset (FICO et al., 2018) used for the Explainable ML Challenge, two recidivism datasets: COMPAS (Larson et al., 2016) and Netherlands (Tollenaar and Van der Heijden, 2013). We predict whether an individual will default on a loan for the FICO dataset, which individuals are arrested within two years of release on the COMPAS dataset, and whether defendants have any type of charge within four years on the Netherlands dataset.

Dataset Name	n	p
Highly Correlated (classification)	800	1000
Highly Correlated (regression)	2000	2000
FICO	10459	1917
COMPAS	6907	134
NETHERLANDS	20000	2024

Table 1: Datasets and their number of samples (n) and number of features (p).

C.2 Data Generation and Preprocessing

Synthetic Datasets

Binary Classification: we generate synthetic datasets according to the generation process in L0Learn (Dedieu et al., 2021). We first sample the data features $\boldsymbol{x}_i \in \mathbb{R}^p$ from a multivariate Gaussian distribution $\mathcal{N}(0, \Sigma)$ with mean 0 and covariance matrix Σ. Then, we create the coefficient vector $\boldsymbol{w}$ with k nonzero entries, where $w_i = 1$ if $i \bmod (p/k) = 0$. Lastly, we sample the data labels $y_i \in \{-1, +1\}$ from a Bernoulli distribution $P(y_i = 1 \mid \boldsymbol{x}_i) = \frac{1}{1+\exp(-\boldsymbol{w}^T \boldsymbol{x}_i)}$. In our experiments, we generate 800 training and 160 test samples with feature dimension $p = 1000$. The data are highly correlated with $\Sigma_{ij} = 0.9^{|i-j|}$. Additionally, we set the number of true sparsity $k = 25$. We generate this setting 5 times with 5 different random seeds (in total we have 5 datasets, each with $(800 + 160) = 960$ samples).

Linear Regression: we generate the synthetic dataset according to the generation process in L0Learn (Hazimeh and Mazumder, 2020) as explained in Section 5.3.1. We first sample the data features $\boldsymbol{x}_i \in \mathbb{R}^p$ from a multivariate Gaussian distribution $\mathcal{N}(0, \Sigma)$ with mean 0 and covariance matrix Σ. Then, we create the coefficient vector $\boldsymbol{w}$ with k nonzero entries, where $w_i = 1$ if $i \bmod (p/k) = 0$. Lastly, we sample $y_i = \boldsymbol{x}_i^T \boldsymbol{w} + \epsilon_i$ with ϵ_i generated from a Gaussian distribution $\mathcal{N}(0, \sigma^2)$. The signal-to-noise ratio (SNR) is defined as $\text{SNR} = \frac{\text{Var}(X\boldsymbol{w})}{\text{Var}(\epsilon)} = \frac{\boldsymbol{w}^T \Sigma \boldsymbol{w}}{\sigma^2}$, where each row of X is $\boldsymbol{x}_i$. In our experiments, we generate 2000 data samples with feature dimension $p = 2000$. The data are highly correlated with $\Sigma_{ij} = 0.9^{|i-j|}$ and $\text{SNR} = 5$. Additionally, we set the true sparsity as $k = 100$.

Real Datasets

FICO: We use all continuous features in this dataset. We did not consider missing data values as separate dummy variables.

COMPAS: We selected features *sex, age, juv_fel_count, juv_misd_count, juv_other_count, priors_count*, and *c_charge_degree* and the label *two_year_recid*.

NETHERLANDS: We translated the feature names from Dutch to English and then used features *sex, country of birth, log # of previous penal cases, 11-20 previous case, and >20 previous case, age in years, age at first penal case, offence type*, and the label *recidivism_in_4y*.

For FICO and COMPAS, we convert each continuous variable $x_{\cdot,j}$ into a set of highly correlated dummy variables $\tilde{x}_{\cdot,j,\theta} = \mathbf{1}_{[x_{\cdot,j} \leq \theta]}$, where θ are all unique values that have appeared in feature column j. For NETHERLANDS, we convert continuous variables into a set of dummy variables in the same way except for variables *age in years* (which is real-valued, not integer) and *age at first penal case*. For these two real-valued variables, instead of considering all unique values that have appeared in the feature column, we consider 1000 quantiles.

C.3 Evaluation Platform

All experimental results were run on a 2.40GHz 30M Cache (256GB RAM 48 hyperthreaded cores) Dell R620 with 2 Xeon(R) CPU E5-2695 v2. We ran all experiments using 8 cores per task.

C.4 Software Packages Used

We list all software packages used in this section. Details about hyperparameter selection are in Appendix D.

- ℓ_1 regularized logistic regression: We run ℓ_1 regularized logistic regression using *glmnet* package (Friedman et al., 2010).
- Minimax Concave Penalty (MCP): We run MCP using *ncvreg* package (Breheny and Huang, 2011).
- L0Learn: We run L0Learn using the R implementation from (Dedieu et al., 2021)[1].
- Ours: We build our method based on L0Learn's codebase, so that we could use its preprocessing steps, and pipeline for running the full regularization path of λ_0 values.

There are some other baselines such as GraSP (Bahmani et al., 2013) and NHTP (Zhou et al., 2021). However, previous work (Dedieu et al., 2021) has shown that they have a considerable number of false positives on the synthetic dataset and have large support sizes for their solutions, so we omit running these two baselines.

C.5 Evaluation Metrics

We use the same evaluation metrics used in Dedieu et al. (2021).

- AUC: The area under the ROC curve.
- Accuracy: $1 - \frac{\sum_{i=1}^{n} \mathbb{1}[y_i \neq \hat{y}_i]}{n}$.
- Recovery F1 score: $\frac{2PR}{P+R}$, where $P = |\text{supp}(\hat{\boldsymbol{w}}) \cap \text{supp}(\boldsymbol{w}^*)| / |\text{supp}(\hat{\boldsymbol{w}})|$ is the precision and $R = |\text{supp}(\hat{\boldsymbol{w}}) \cap \text{supp}(\boldsymbol{w}^*)| / |\text{supp}(\boldsymbol{w}^*)|$ is the recall. $\text{supp}(\cdot)$ stands for the support (indices with nonzero coefficients) of a solution. We can only use recovery F1 score for synthetic datasets since we need to know the support of $\boldsymbol{w}^*$ to calculate it.

D ADDITIONAL EXPERIMENTS

We first elaborate on hyperparameters used for different software packages. We then present extra experimental results that were omitted from the main paper due to space constraints.

Collection and Setup: we ran the experiments on the two simulated datasets and 3 real datasets: FICO, COMPAS, and Netherlands. For each dataset, we trained the model using varying configurations. On the simulated classification task, we ran on 5 datasets, each generated by a different random seed. On the real datasets, we performed 5-fold cross validation to measure training time, training accuracy, and test accuracy for each fold.

To get the support versus AUC, accuracy, and F1 score curves, for MCP, the sequence of 100 λ values was set to the default values of ncvreg, and we chose the second parameter γ by using 10 values between 1.5 and 25, where we show results of γ being 1.5 and 25 for each λ. Curves for other γ values are between these extremes. For ℓ_1 regularized logistic regression, the choice of 100 λ values was set to the default sequence chosen by glmnet. For L0Learn, we set the penalty type to "L0L2" and γ to $\{0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10\}$. The regularization

[1]https://github.com/hazimehh/L0Learn

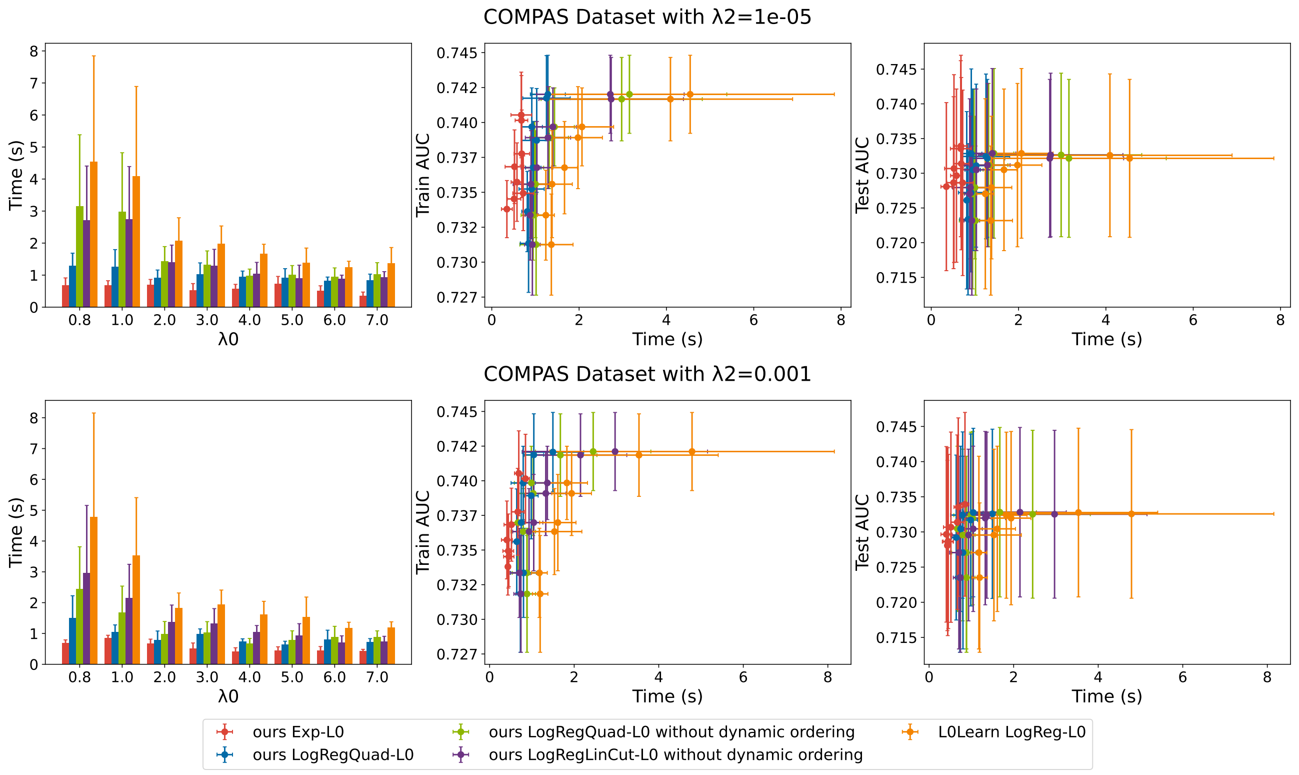

Figure 6: Computational times of different methods. "Exp" stands for exponential loss, "LogReg" stands for logistic loss, and "Quad" stands for quadratic cuts. Note that there is no ℓ_2 penalty for the exponential loss. *Our Exp-L0 method is generally about 4 times faster than L0Learn.* Note that the AUC axes indicate practically similar performance for these particular methods; the training time is what differentiates the methods.

choices for the ℓ_0 term were set to the 100 default values of L0Learn. For our methods, we also set the penalty type and γ (which is λ_2) in the same way as the setting for L0Learn and use the same λ values as in the L0Learn algorithm.

In addition, we set 56 pairs of λ (resp. λ_0) and γ (resp. λ_2) values for comparing the run times obtained by our methods and by L0Learn: $\lambda \in \{0.8, 1, 2, 3, 4, 5, 6, 7\}$ and $\gamma \in \{0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10\}$.

D.1 Run Time Savings for Linear Regression

Although our method is designed for classification problems, the proposed dynamic ordering technique can also speed up the local swap process for linear regression. For the full regularization path with 100 different λ_0 values, the total time difference between local swaps without dynamic ordering and local swaps with dynamic ordering improved computation time by 36% (from 184 seconds to 117 seconds).

D.2 Run Time Savings from First and Second Methods

We show results on time savings from our first method (linear cut, quadratic cut, and dynamic ordering) and our second method (exponential loss). The general trends are: *i*) Using the quadratic cut makes the algorithm faster than the linear cut, i.e., there is more time saved with stronger ℓ_2 regularization. *ii*) Using dynamic ordering and the quadratic cut together makes the algorithm much faster than using the quadratic cut alone. *iii*) When features are binary, using the exponential loss has the greatest computational advantage. These trends are shown fairly uniformly across datasets. Results for each dataset are shown in Figures 6-7.

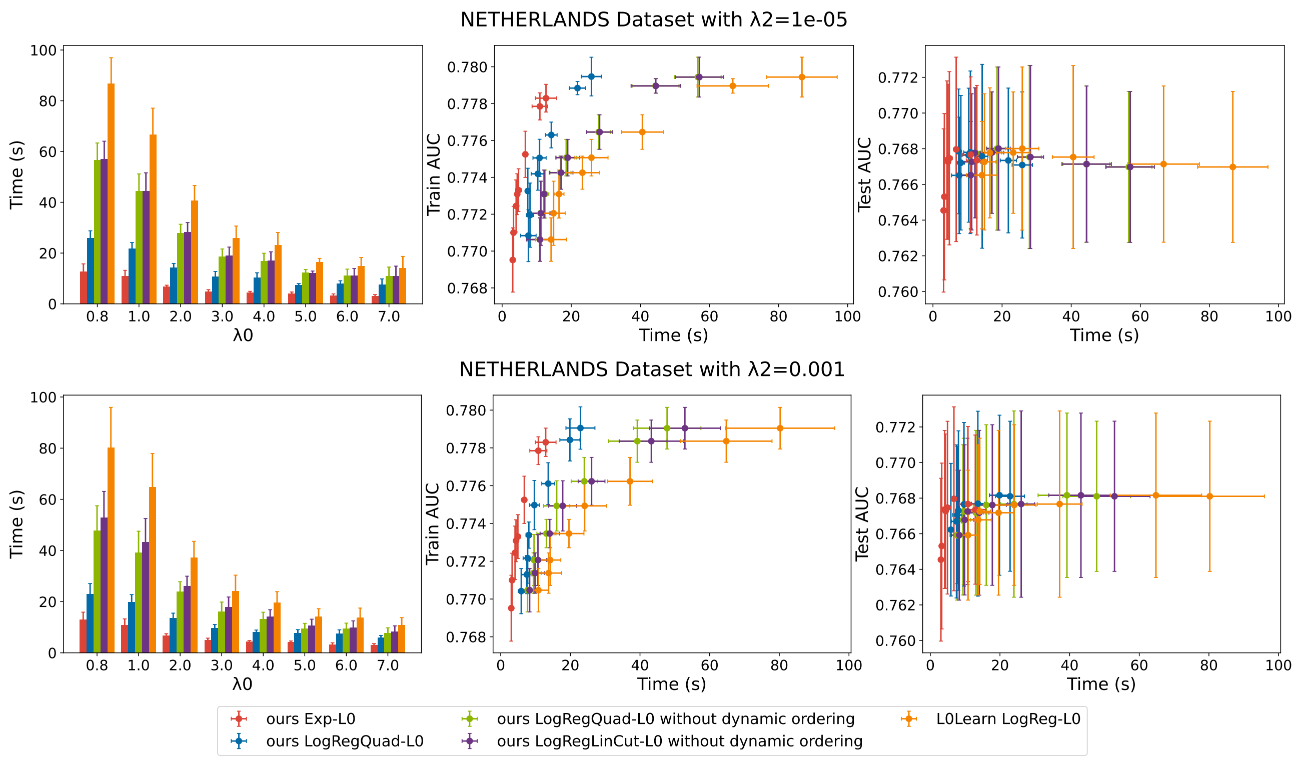

Figure 7: Computational times of different methods. "Exp" stands for exponential loss, "LogReg" stands for logistic loss, and "Quad" stands for quadratic cuts. Note that there is no ℓ_2 penalty for the exponential loss. *Our Exp-L0 method is generally about 4 times faster than L0Learn.* Note that the AUC axes indicate practically similar performance for these particular methods; the training time is what differentiates the methods.

D.3 Support versus AUC, Accuracy, and F1 Score

We provide the full regularization paths for the FICO dataset (see Figure 8). Our first method (quadratic cut + dynamic ordering) and second method (exponential loss) obtain high-quality solutions and their AUC and accuracy curves are similar to those from other methods. Our methods have computational advantages over L0Learn, as shown in Section D.2.

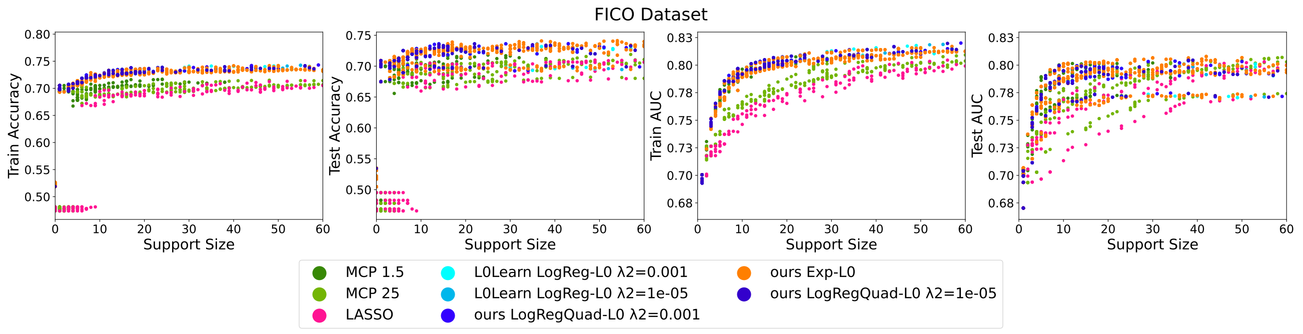

Figure 8: Results on the FICO dataset. See Figure 5 in the main paper for results on the COMPAS and NETHERLANDS datasets. The L0Learn points are mostly overlapping with points from our methods.

To investigate whether a small ℓ_2 regularization would help with the LASSO baseline, we provide a comparison between our method and the the ElasticNet method on the highly correlated synthetic dataset for the classification task. We used the glmnet R package for the ElasticNet baseline. The hyperparameter $\alpha \in [0, 1]$ controls the balance between ℓ_1 and ℓ_2 regularization. The LASSO method corresponds to $\alpha = 1.0$. Besides $\alpha = 1.0$, we also consider $\alpha \in \{0.9, 0.7, 0.5, 0.3, 0.1, 0.001\}$. As shown in Figure 9, a small ℓ_2 regularization term does not improve the LASSO method much. When α decreases, the solution quality degrades. This is potentially because more ℓ_2 regularization leads to non-sparse solutions with small coefficients, neither of which will lead to better performance here.

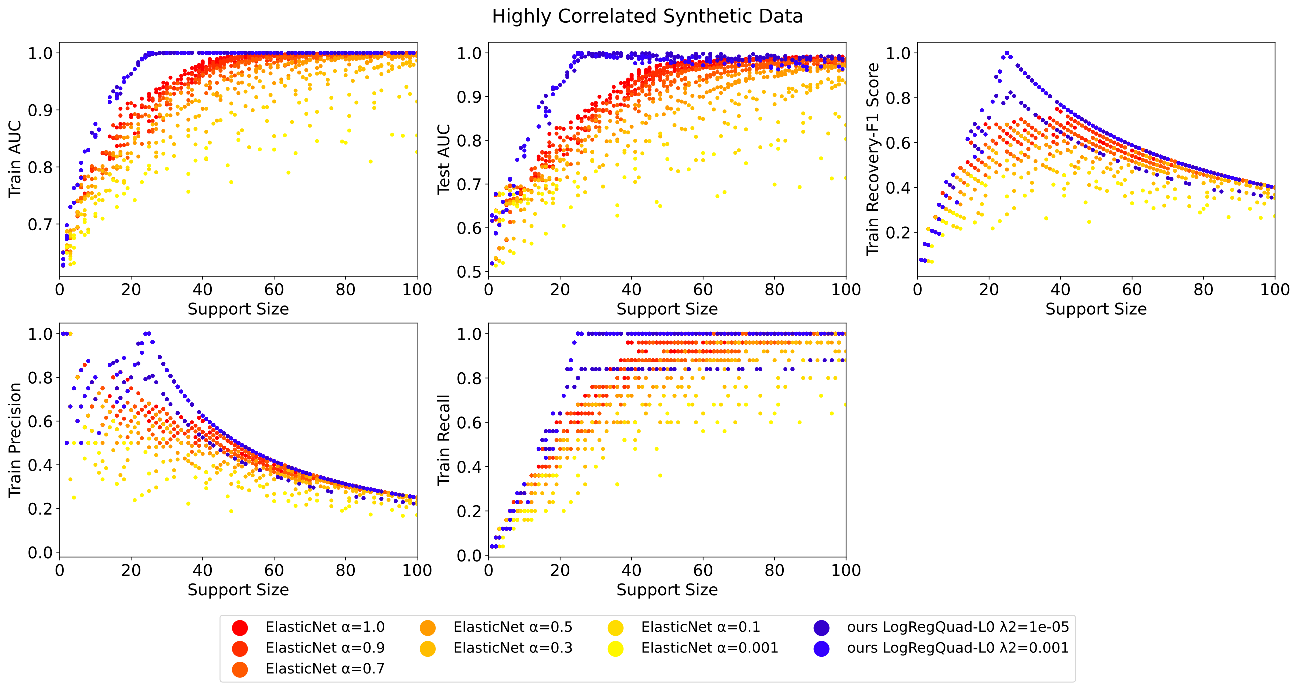

Figure 9: ElasticNet and our methods' results on the highly correlated synthetic dataset.

D.4 Samples of Sparse Models on the FICO and NETHERLANDS datasets

We provide some sample sparse models produced by minimizing the exponential loss and minimizing the logistic loss (quadratic cut + dynamic ordering) on the FICO and NETHERLANDS datasets.

The FICO dataset has 10459 samples and 1917 features. The NETHERLANDS dataset has 20000 samples and 2024 features. All models were developed from the third fold of our 5-fold cross validation split.

FICO Baseline Performance: The sparse models below approximately match the performance of black-box models shown in previous works (Chen et al., 2021). We also ran a GBDT model (Friedman, 2001) with max depth set to be 3 and number of boosting stages set to be 100. The AUC on the training set is 0.8318 ± 0.0028, and the AUC on the test set is 0.7959 ± 0.0133. The result on the test set is comparable to what we have shown in Figure 3. The models are in Figures 10-11.

NETHERLANDS Baseline Performance: The sparse models below approximately match the performance of black-box models. We ran a GBDT model (Friedman, 2001) with max depth set to be 3 and number of boosting stages set to be 100. The AUC on the training set is 0.7850 ± 0.0014, and the AUC on the test set is 0.7696 ± 0.0062. The result on the test set is comparable to what we have shown in Figure 7. (For the NETHERLANDS dataset, ages are collected in terms of months. That is why age thresholds are shown with float numbers.) The models are in Figures 12-13.

FICO model using the exponential loss:

$\lambda_0 = 5$:

$$\begin{aligned}
score = &- 0.2584626 \\
&+ 0.1825955 \times \mathbf{1}_{A \leq 63} + 0.1387806 \times \mathbf{1}_{A \leq 70} \\
&+ 0.2286364 \times \mathbf{1}_{A \leq 74} + 0.2569742 \times \mathbf{1}_{A \leq 83} && \# \ A \ \text{:}ExternalRiskEstimate \\
&+ 0.1840013 \times \mathbf{1}_{B \leq 51} + 0.172138 \times \mathbf{1}_{B \leq 75} && \# \ B \ \text{:}AverageMInFile \\
&+ 0.2015039 \times \mathbf{1}_{C \leq 13} + 0.1923697 \times \mathbf{1}_{C \leq 31} && \# \ C \ \text{:}NumSatisfactoryTrades \\
&+ 0.2654667 \times \mathbf{1}_{D \leq 96} && \# \ D \ \text{:}PercentTradesNeverDelq \\
&+ 0.2320259 \times \mathbf{1}_{E \leq 33} && \# \ E \ \text{:}MSinceMostRecentDelq \\
&+ 0.1009372 \times \mathbf{1}_{F \leq 8} && \# \ F \ \text{:}NumTotalTrades \\
&- 0.2311165 \times \mathbf{1}_{G \leq 46} && \# \ G \ \text{:}PercentInstallTrades \\
&- 0.7723769 \times \mathbf{1}_{H \leq -8} + 0.3636577 \times \mathbf{1}_{H \leq 0} && \# \ H \ \text{:}MSinceMostRecentInqexcl7days \\
&- 0.2762694 \times \mathbf{1}_{I \leq 5} && \# \ I \ \text{:}NumInqLast6M \\
&- 0.1897788 \times \mathbf{1}_{J \leq 37} - 0.2742168 \times \mathbf{1}_{J \leq 73} && \# \ J \ \text{:}NetFractionRevolvingBurden \\
&- 0.1038025 \times \mathbf{1}_{K \leq 5} - 0.1938047 \times \mathbf{1}_{K \leq 7} && \# \ K \ \text{:}NumRevolvingTradesWBalance
\end{aligned}$$

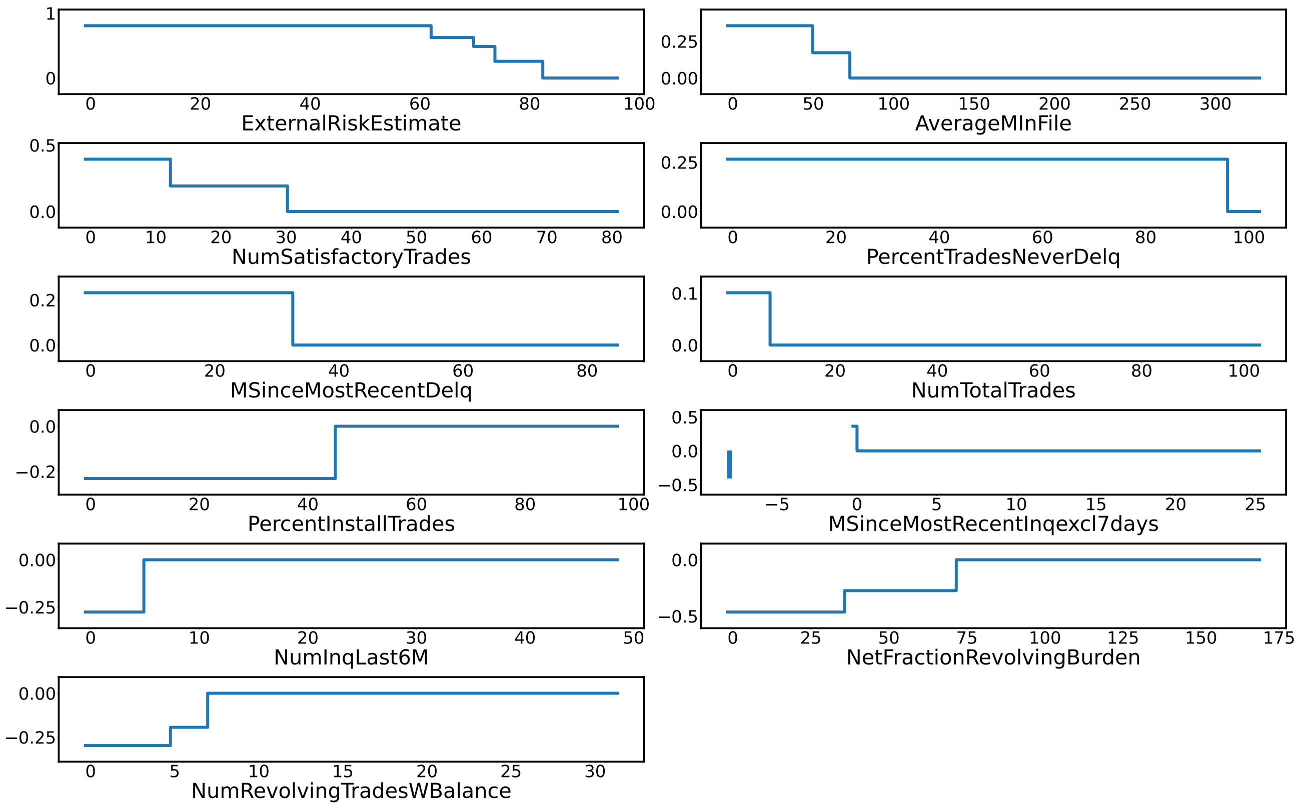

Figure 10: FICO score contributions with the exponential loss and $\lambda_0 = 5$. Training duration is 3.15 seconds. Note that no monotonicity constraints were imposed.

FICO model using the logistic loss (quadratic cut + dynamic ordering):
$\lambda_0 = 5$, $\lambda_2 = 0.001$:

$$
\begin{aligned}
score = & 2.805021 \\
& + 0.4071199 \times \mathbf{1}_{A \le 63} + 0.310368 \times \mathbf{1}_{A \le 70} \\
& + 0.4604512 \times \mathbf{1}_{A \le 74} + 0.5471219 \times \mathbf{1}_{A \le 83} && \# \ A: \mathit{ExternalRiskEstimate} \\
& + 0.408959 \times \mathbf{1}_{B \le 51} + 0.3283239 \times \mathbf{1}_{B \le 75} && \# \ B: \mathit{AverageMInFile} \\
& + 0.4225237 \times \mathbf{1}_{C \le 13} + 0.3396898 \times \mathbf{1}_{C \le 31} && \# \ C: \mathit{NumSatisfactoryTrades} \\
& + 0.525166 \times \mathbf{1}_{D \le 96} && \# \ D: \mathit{PercentTradesNeverDelq} \\
& + 0.4427697 \times \mathbf{1}_{E \le 33} && \# \ E: \mathit{MSinceMostRecentDelq} \\
& - 0.4317725 \times \mathbf{1}_{F \le 46} && \# \ F: \mathit{PercentInstallTrades} \\
& - 1.576435 \times \mathbf{1}_{G \le -8} + 0.5045199 \times \mathbf{1}_{G \le 0} \\
& + 0.2874494 \times \mathbf{1}_{G \le 1} && \# \ G: \mathit{MSinceMostRecentInqexcl7days} \\
& - 3.97116 \times \mathbf{1}_{H \le 11} && \# \ H: \mathit{NumInqLast6M} \\
& - 0.3657186 \times \mathbf{1}_{I \le 37} - 0.5681891 \times \mathbf{1}_{I \le 73} && \# \ I: \mathit{NetFractionRevolvingBurden} \\
& - 0.4969551 \times \mathbf{1}_{J \le 7} && \# \ J: \mathit{NumRevolvingTradesWBalance}
\end{aligned}
$$

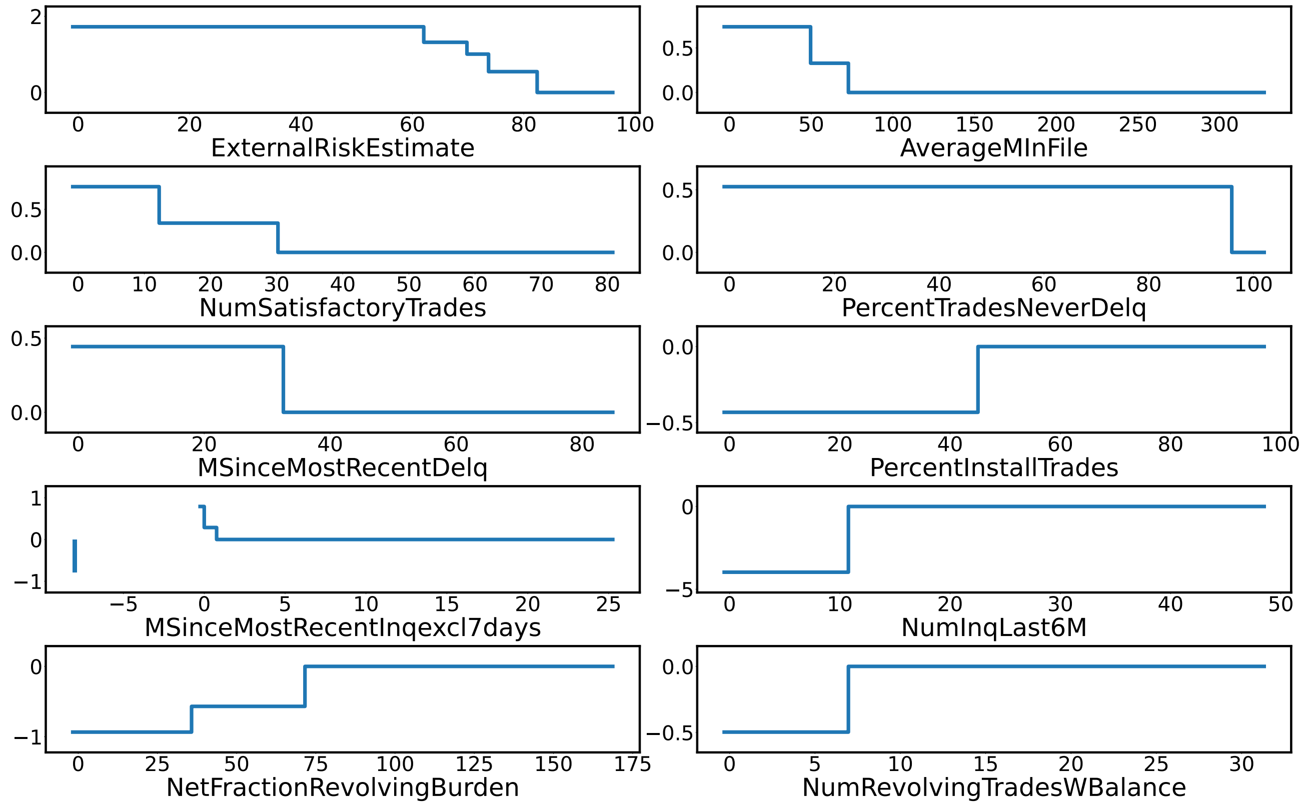

Figure 11: FICO score contributions with the logistic loss and $\lambda_0 = 5, \lambda_2 = 0.001$. Training duration is 5.95 seconds.

NETHERLANDS model using the exponential loss:

$\lambda_0 = 7$:

$$
\begin{aligned}
score = & 3.114342 \\
& - 0.1439394 \times \mathbf{1}_{A==female} && \# \ A\ :sex \\
& - 0.4739628 \times \mathbf{1}_{B\leq 0.0} - 0.3336059 \times \mathbf{1}_{B\leq 1.098612289} \\
& - 0.3083761 \times \mathbf{1}_{B\leq 1.609437912} && \# \ B\ :log\ \#\ of\ previous\ penal\ cases \\
& + 0.2887266 \times \mathbf{1}_{C\leq 22.26146475} + 0.2354507 \times \mathbf{1}_{C\leq 28.48266213076} \\
& + 0.1951787 \times \mathbf{1}_{C\leq 39.07432555374} + 0.2304243 \times \mathbf{1}_{C\leq 46.91581109} && \# \ C\ :age\ in\ years \\
& - 0.1674992 \times \mathbf{1}_{D\leq 28.069209540720003} && \# \ D\ :age\ at\ first\ penal\ case \\
& + 0.1526613 \times \mathbf{1}_{E\leq 7.0} && \# \ E\ :offence\ type \\
& - 1.381801 \times \mathbf{1}_{F\leq 0.0} && \# \ F\ :11\text{-}20\ previous\ case \\
& - 1.856642 \times \mathbf{1}_{G\leq 0.0} && \# \ G\ :>20\ previous\ case
\end{aligned}
$$

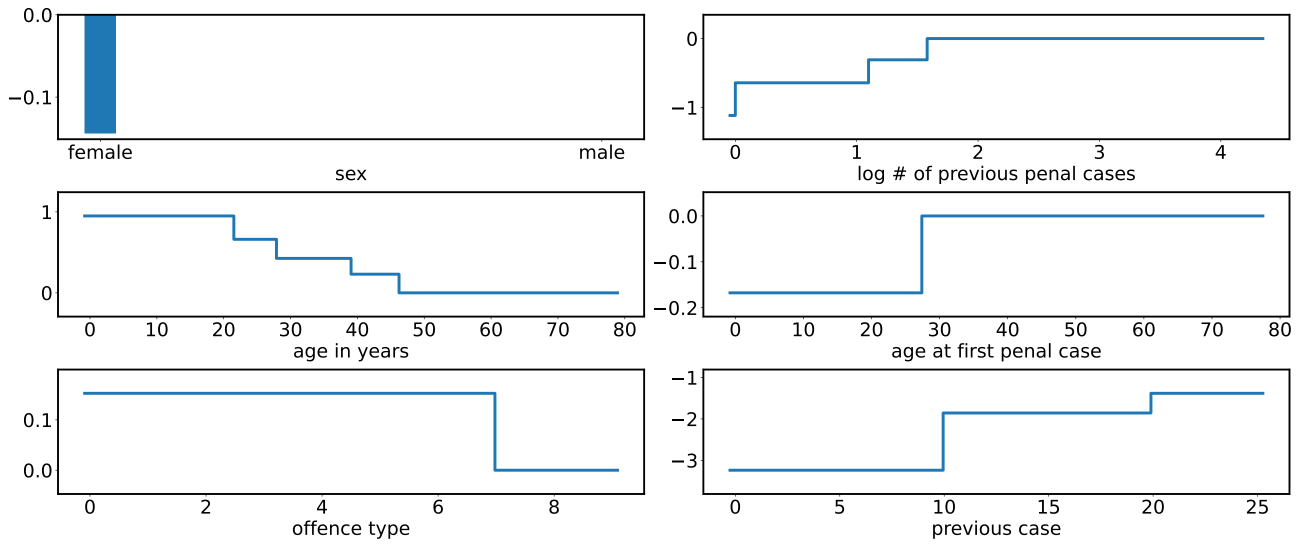

Figure 12: NETHERLANDS score contributions with the exponential loss and $\lambda_0 = 7$. Training duration is 2.73 seconds. Note that there are no monotonicity constraints imposed.

NETHERLANDS model using the logistic loss (quadratic cut + dynamic ordering):
$\lambda_0 = 7$, $\lambda_2 = 0.001$:

$$
\begin{aligned}
score = & 6.259994 \\
& - 0.3092142 \times \mathbf{1}_{A==female} && \# \ A \ :sex \\
& - 0.7374188 \times \mathbf{1}_{B\le 0.0} - 0.4302188 \times \mathbf{1}_{B\le 0.693147181} \\
& - 0.2888496 \times \mathbf{1}_{B\le 1.098612289} - 0.3933033 \times \mathbf{1}_{B\le 1.386294361} \\
& - 0.5383587 \times \mathbf{1}_{B\le 1.945910149} && \# \ B \ :log \ \# \ of \ previous \ penal \ cases \\
& + 0.3877289 \times \mathbf{1}_{C\le 18.94046991832} + 0.5554352 \times \mathbf{1}_{C\le 23.01017483608} \\
& + 0.4700141 \times \mathbf{1}_{C\le 31.552317465359998} + 0.6324188 \times \mathbf{1}_{C\le 43.91512663} && \# \ C \ :age \ in \ years \\
& - 0.2645467 \times \mathbf{1}_{D\le 27.986380572549994} && \# \ D \ :age \ at \ first \ penal \ case \\
& + 0.2861914 \times \mathbf{1}_{E\le 7.0} && \# \ E \ :offence \ type \\
& - 2.655844 \times \mathbf{1}_{F\le 0.0} && \# \ F \ :11\text{-}20 \ previous \ case \\
& - 3.605789 \times \mathbf{1}_{G\le 0.0} && \# \ G \ :>20 \ previous \ case
\end{aligned}
$$

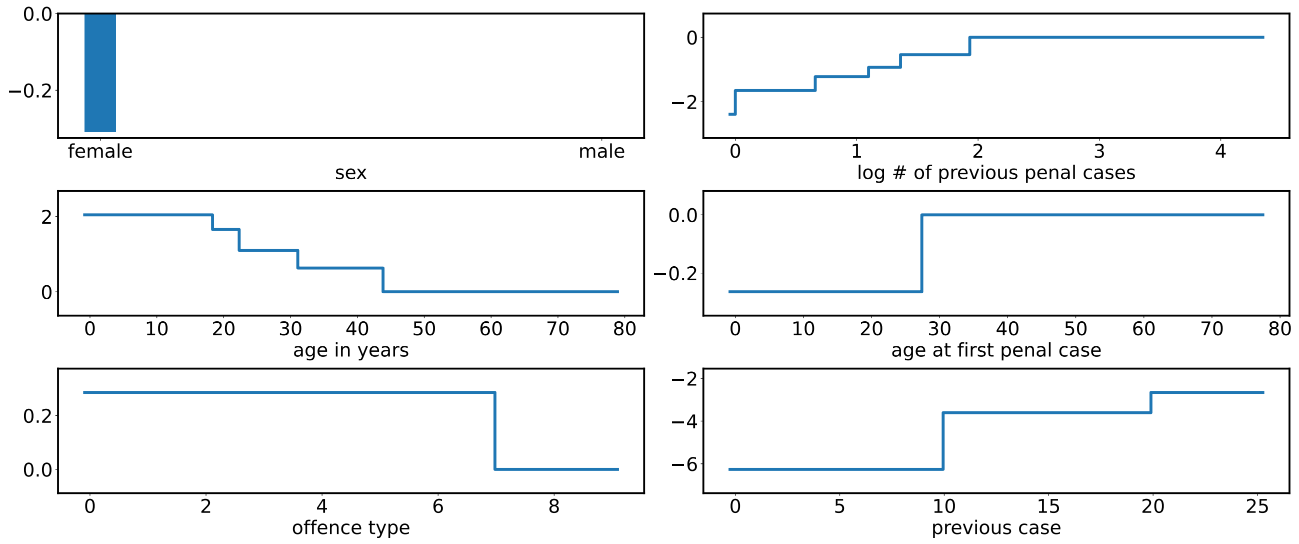

Figure 13: NETHERLANDS model score contributions with logistic loss, $\lambda_0 = 7$, and $\lambda_2 = 0.001$. Training duration is 7.2 seconds.