Appendices for WHInter

A. Proof of Lemma 3.1

Lemma 3.1. *For any* $\mathbf{X} \in \{0,1\}^{n\times p}$, $\mathbf{v} \in \mathbb{R}^n_+$, $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \in \mathbb{R}^n$, $j \in [\![p]\!]$, $\mathcal{I} \subset [\![p]\!]$ *and* $\alpha \in \mathbb{R}$, *the following holds:*

$$\max_{k\in\mathcal{I}} \left| \boldsymbol{\theta}_2^\top (\mathbf{v} \odot \mathbf{X}_k) \right| \leq |\,\alpha\,| \max_{k\in\mathcal{I}} \left| \boldsymbol{\theta}_1^\top (\mathbf{v} \odot \mathbf{X}_k) \right| + \zeta(\boldsymbol{\theta}_2 - \alpha\boldsymbol{\theta}_1, \mathbf{v})\,, \tag{1}$$

where

$$\forall (\mathbf{u}, \mathbf{v}) \in \mathbb{R}^n \times \mathbb{R}^n_+\,, \quad \zeta(\mathbf{u}, \mathbf{v}) = \max\left(\sum_{i:\mathbf{u}_i>0} \mathbf{u}_i\mathbf{v}_i, -\sum_{i:\mathbf{u}_i<0} \mathbf{u}_i\mathbf{v}_i \right).$$

Proof. With the notations of Lemma 3.1, we have:

$$\begin{aligned}
\max_{k\in\mathcal{I}} \left| \boldsymbol{\theta}_2^\top (\mathbf{v} \odot \mathbf{X}_k) \right| &\leq \max_{k\in\mathcal{I}} \left| \alpha\boldsymbol{\theta}_1^\top (\mathbf{v} \odot \mathbf{X}_k) + (\boldsymbol{\theta}_2 - \alpha\boldsymbol{\theta}_1)^\top (\mathbf{v} \odot \mathbf{X}_k) \right| \\
&\leq |\,\alpha\,| \max_{k\in\mathcal{I}} \left| \boldsymbol{\theta}_1^\top (\mathbf{v} \odot \mathbf{X}_k) \right| + \max_{k\in\mathcal{I}} \left| (\boldsymbol{\theta}_2 - \alpha\boldsymbol{\theta}_1)^\top (\mathbf{v} \odot \mathbf{X}_k) \right| \\
&\leq |\,\alpha\,| \max_{k\in\mathcal{I}} \left| \boldsymbol{\theta}_1^\top (\mathbf{v} \odot \mathbf{X}_k) \right| + \max_{\boldsymbol{x}\in\{0,1\}^n} \left| (\boldsymbol{\theta}_2 - \alpha\boldsymbol{\theta}_1)^\top (\mathbf{v} \odot \boldsymbol{x}) \right| \\
&= |\,\alpha\,| \max_{k\in\mathcal{I}} \left| \boldsymbol{\theta}_1^\top (\mathbf{v} \odot \mathbf{X}_k) \right| + \zeta(\boldsymbol{\theta}_2 - \alpha\boldsymbol{\theta}_1, \mathbf{v})\,.
\end{aligned}$$

□

B. Computing η_{min}

In this section we characterise the existence and the possibility to compute, for any fixed $(\mathbf{v}, \boldsymbol{\theta}, \boldsymbol{\theta}', m) \in \mathbb{R}^n_+ \times \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}$:

$$\eta_{min}\left(\mathbf{v}, \boldsymbol{\theta}, \boldsymbol{\theta}', m\right) := \min_{\alpha\in\mathbb{R}} \eta_\alpha\left(\mathbf{v}, \boldsymbol{\theta}, \boldsymbol{\theta}', m\right)\,, \tag{2}$$

where η_α is defined in Section 3.2. For that purpose, let us introduce for any $\alpha \in \mathbb{R}$ the functions:

$$\begin{cases}
\gamma_p(\alpha) &= \displaystyle\sum_{i:\boldsymbol{\theta}'_i - \alpha\boldsymbol{\theta}_i>0} \mathbf{v}_i \left(\boldsymbol{\theta}'_i - \alpha\boldsymbol{\theta}_i\right), \\
\gamma_m(\alpha) &= \displaystyle\sum_{i:\boldsymbol{\theta}'_i - \alpha\boldsymbol{\theta}_i<0} \mathbf{v}_i \left(\boldsymbol{\theta}'_i - \alpha\boldsymbol{\theta}_i\right),
\end{cases}$$

such that:

$$\eta_\alpha\left(\mathbf{v}, \boldsymbol{\theta}, \boldsymbol{\theta}', m\right) = |\alpha|\, m + \max\left(\gamma_p(\alpha), -\gamma_m(\alpha)\right)\,. \tag{3}$$

Let us first characterise the existence and properties of the solution to the minimisation problem (2).

Theorem S1. *For any* $(\mathbf{v}, \boldsymbol{\theta}, \boldsymbol{\theta}', m) \in \mathbb{R}_+^n \times \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}$*, the function*

$$\alpha \in \mathbb{R} \to \eta_\alpha\left(\mathbf{v}, \boldsymbol{\theta}, \boldsymbol{\theta}', m\right)$$

is continuous, piecewise affine, convex and nonnegative. It reaches at least a minimum at a value $\alpha^* \in \mathcal{B}$ *where*

$$\mathcal{B} = \{0\} \cup \left\{ \frac{\boldsymbol{\theta}'_i}{\boldsymbol{\theta}_i} \,:\, i \in \mathit{supp}(\boldsymbol{\theta}) \cap \mathit{supp}(\mathbf{v}) \right\} \cup \{\alpha \in \mathbb{R} : \gamma_p(\alpha) = \gamma_m(\alpha)\} \,.$$

Proof. For any $i \in [\![n]\!]$, let

$$\forall \alpha \in \mathbb{R}\,, \quad \phi_i(\alpha) = \mathbf{v}_i \max\left(0, \boldsymbol{\theta}'_i - \alpha \boldsymbol{\theta}_i\right) \,.$$

Since $\mathbf{v}_i \geq 0$, $\phi_i(\alpha) = \mathbf{v}_i \max(0, \boldsymbol{\theta}'_i - \alpha\boldsymbol{\theta}_i)$ is continuous, piecewise affine, convex and nonnegative. It has a single breakpoint at $\alpha_i = \boldsymbol{\theta}'_i/\boldsymbol{\theta}_i$ if $\boldsymbol{\theta}_i \neq 0$ and $\mathbf{v}_i > 0$, and is constant otherwise. Since $\gamma_p(\alpha) = \sum_{i=1}^n \phi_i(\alpha)$, γ_p is also continuous, piecewise affine, convex and nonnegative with breakpoints in $\{\boldsymbol{\theta}'_i/\boldsymbol{\theta}_i \,:\, i \in \mathrm{supp}(\boldsymbol{\theta}) \cup \mathrm{supp}(\mathbf{v})\}$. Taking $\psi_i(\alpha) = \mathbf{v}_i \max(0, \alpha\boldsymbol{\theta}_i - \boldsymbol{\theta}'_i)$ shows similarly that $-\gamma_m(\alpha) = \sum_{i=1}^n \psi_i(\alpha)$ has the same properties. Consequently, the function $\alpha \mapsto \max(\gamma_p(\alpha), -\gamma_m(\alpha))$ is also continuous, piecewise affine, convex and nonnegative, with possible breakpoints in

$$\left\{\boldsymbol{\theta}'_i/\boldsymbol{\theta}_i \,:\, i \in \mathrm{supp}(\boldsymbol{\theta}) \cup \mathrm{supp}(\mathbf{v})\right\} \cup \{\alpha \in \mathbb{R} : \gamma_p(\alpha) = \gamma_m(\alpha)\} \,.$$

Since $\alpha \mapsto |\,\alpha\,|$ is also continuous, piecewise affine, convex and nonnegative, and has a breakpoint for $\alpha = 0$, Theorem S1 follows by observing that a continuous, piecewise affine, convex and nonnegative function necessarily reaches a minimum at one of its breakpoints. □

Let $S = |\,\mathrm{supp}(\boldsymbol{\theta}) \cap \mathrm{supp}(\mathbf{v})\,|$. Theorem S1 shows that it suffices to compute the values of η_α on at most $S + 2$ values for α to find the global minimum. However, a naive computation of η_α using (3) takes $O(|\,\mathrm{supp}(\mathbf{v})\,|)$ for each α, hence a total complexity $O(S \times |\,\mathrm{supp}(\mathbf{v})\,|)$ to find the minimum of η_α.

This can be improved to $O(|\,\mathrm{supp}(\mathbf{v})\,| + S \ln S)$ by first sorting the $S+1$ breakpoints $b_i = \boldsymbol{\theta}'_i/\boldsymbol{\theta}_i$ for $i \in \mathrm{supp}(\boldsymbol{\theta}) \cap \mathrm{supp}(\mathbf{v})$ and $b_{S+1} = 0$ in increasing order:

$$b_{\pi(1)} \leq b_{\pi(2)} \leq \ldots \leq b_{\pi(S+1)} \,,$$

which takes $O(S \ln S)$ time. Adding by convention $b_{\pi(0)} = -\infty$ we observe that on each interval $(b_{k-1}, b_k]$ the functions γ_p and γ_m are affine, of the form:

$$\forall \alpha \in (b_{k-1}, b_k]\,, \quad \begin{cases} \gamma_p(\alpha) & = s_p^k - \alpha t_p^k \,, \\ -\gamma_m(\alpha) & = s_m^k - \alpha t_m^k \,. \end{cases}$$

From the properties $\gamma_p(\alpha) = \sum_{i=1}^n \phi_i(\alpha)$ and $-\gamma_m(\alpha) = \sum_{i=1}^n \psi_i(\alpha)$, we get the coefficients for $k = 1$, i.e., the interval $(-\infty, b_{\pi(1)}]$, in $O(|\,\mathrm{supp}(\mathbf{v})\,|)$ as follows:

$$\begin{cases} s_p^1 & = \sum_{i \in \mathrm{supp}(\mathbf{v}) \,:\, \boldsymbol{\theta}_i > 0} \mathbf{v}_i \boldsymbol{\theta}'_i + \sum_{i \in \mathrm{supp}(\mathbf{v}) \,:\, \boldsymbol{\theta}_i = 0} \mathbf{v}_i \max(0, \boldsymbol{\theta}'_i) \,, \\ t_p^1 & = \sum_{i \in \mathrm{supp}(\mathbf{v}) \,:\, \boldsymbol{\theta}_i > 0} \mathbf{v}_i \boldsymbol{\theta}_i \,, \\ s_m^1 & = -\sum_{i \in \mathrm{supp}(\mathbf{v}) \,:\, \boldsymbol{\theta}_i < 0} \mathbf{v}_i \boldsymbol{\theta}'_i + \sum_{i \in \mathrm{supp}(\mathbf{v}) \,:\, \boldsymbol{\theta}_i = 0} \mathbf{v}_i \max(0, -\boldsymbol{\theta}'_i) \,, \\ t_m^1 & = \sum_{i \in \mathrm{supp}(\mathbf{v}) \,:\, \boldsymbol{\theta}_i < 0} \mathbf{v}_i \boldsymbol{\theta}_i \,. \end{cases} \tag{4}$$

This allows in particular to compute $\gamma_p(b_{\pi(1)})$, $\gamma_m(b_{\pi(1)})$, and therefore $\eta_{b_{\pi(1)}}$ from (3). We can then iteratively compute the coefficients for $k+1$ from the coefficients for k in $O(1)$ only, by observing that between the intervals $(b_{k-1}, b_k]$ and $(b_k, b_{k+1}]$, the only change in slope and intercept of γ_p is due to the function $\phi_{\pi^{-1}(k)}$, when $\pi^{-1}(k) \neq S+1$. Let $i = \pi^{-1}(k)$. When $\boldsymbol{\theta}_i > 0$, the slope of ϕ_i increases by $\mathbf{v}_i\boldsymbol{\theta}_i$ and its intercept decreases by $\mathbf{v}_i\boldsymbol{\theta}'_i$ at b_i. When $\boldsymbol{\theta}_i > 0$, its slope increases by $-\mathbf{v}_i\boldsymbol{\theta}_i$ and its intercept increases by $\mathbf{v}_i\boldsymbol{\theta}'_i$. This translates into the following recursive formula for the coefficients of γ_p:

$$s_p^{k+1} = \begin{cases} s_p^k - \mathbf{v}_i\boldsymbol{\theta}'_i & \text{if } \boldsymbol{\theta}_i > 0\,, \\ s_p^k + \mathbf{v}_i\boldsymbol{\theta}'_i & \text{if } \boldsymbol{\theta}_i < 0\,, \end{cases}$$

and

$$t_p^{k+1} = t_p^k - \mathbf{v}_i \,|\, \boldsymbol{\theta}_i \,|\,.$$

A similar analysis on γ_m leads to the following recursion:

$$s_m^{k+1} = \begin{cases} s_m^k - \mathbf{v}_i\boldsymbol{\theta}'_i & \text{if } \boldsymbol{\theta}_i > 0\,, \\ s_m^k + \mathbf{v}_i\boldsymbol{\theta}'_i & \text{if } \boldsymbol{\theta}_i < 0\,, \end{cases}$$

and

$$t_m^{k+1} = t_m^k - \mathbf{v}_i \,|\, \boldsymbol{\theta}_i \,|\,.$$

We can thus iteratively compute the coefficients on each interval, and thus the values of η_α on each breakpoint, with complexity $O(1)$ per breakpoint. Since $\alpha \mapsto \eta_\alpha$ is convex, we stop at the first k such that $\eta_{b_{\pi(k+1)}} \geq \eta_{b_{\pi(k)}}$. From the equations of γ_p and γ_m on $(b_{\pi(k)}, b_{\pi(k+1)}]$ we can additionally check if there is a crossing point $\bar{\alpha} \in (b_{\pi(k)}, b_{\pi(k+1)}]$ such that $\gamma_p(\bar{\alpha}) = \gamma_m(\bar{\alpha})$, in which case we also compute $\eta_{\bar{\alpha}}$. The global minimum of $\alpha \mapsto \eta_\alpha$ is then $\min(\eta_{b_{\pi(k)}}, \eta_{\bar{\alpha}})$.

The overall algorithm is detailed in Algorithm S1.

C. Alternative solver for working set updates

In this section, we present an alternative solver to the inverted list approach (Algorithm 3 in Section 3.3), which we call $MIPS1$, to compute the working set updates (defined in Section 3.3). It relies on a pruning technique and does not require storing extra indices for the data. The main idea of this alternative approach is to compute inner products on a progressively growing subset of dimensions, and to maintain an upper-bound on the maximum attainable score on the remaining dimensions. This allows to discard a probe as soon as its maximum attainable score drops below the maximum score achieved so far without computing the inner product in its entirety. Algorithm S2 presents the procedure in details. It takes as input $\mathcal{Q}$ which contains the indices that define the queries of interest and outputs the updated working set $\mathcal{W}$ and $\mathbf{m}^{ref}$. For each query, we start by precomputing the partial inner product bounds $\mathbf{r}^+ \in \mathbb{R}^n$ and $\mathbf{r}^- \in \mathbb{R}^n$, where $\mathbf{r}_i^+$ and $\mathbf{r}_i^-$ are respectively the maximum and minimum attainable inner products between the query and any probe in the database on the dimensions from $i+1$ to n. Formally, $\mathbf{r}^+$ and $\mathbf{r}^-$ are defined for a given query j by:

$$\forall i \in [\![n]\!]\,, \boldsymbol{r}_i^+ = \sum_{m>i;\ \boldsymbol{\theta}_m>0} \mathbf{X}_{mj}\boldsymbol{\theta}_m \tag{5}$$

$$\forall i \in [\![n]\!]\,, \boldsymbol{r}_i^- = \sum_{m>i;\ \boldsymbol{\theta}_m<0} \mathbf{X}_{mj}\boldsymbol{\theta}_m \tag{6}$$

Algorithm S1 Minimise η in α

Input: $(\mathbf{v}, \boldsymbol{\theta}, \boldsymbol{\theta}', m) \in \mathbb{R}_+^n \times \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}$.

Output: $\eta_{min}(\mathbf{v}, \boldsymbol{\theta}, \boldsymbol{\theta}', m)$

$S \leftarrow$ indices in $\text{supp}(\mathbf{v}) \cap \text{supp}(\boldsymbol{\theta})$
$N \leftarrow \text{length}(S)$
$b \leftarrow \left[0, \frac{\boldsymbol{\theta}'_{S[1]}}{\boldsymbol{\theta}_{S[1]}}, \ldots, \frac{\boldsymbol{\theta}'_{S[N]}}{\boldsymbol{\theta}_{S[N]}}\right]$
$\texttt{ind} \leftarrow [\texttt{none}, S[1], \ldots, S[N]]$
$\texttt{rank} \leftarrow \text{sort}(b)$ (in increasing order)
$b \leftarrow b[\texttt{rank}]$; $\texttt{ind} \leftarrow \texttt{ind}[\texttt{rank}]$
Initialise s_p, s_m, t_p, t_m via (4)
$\texttt{min} \leftarrow +\infty$
for i in $1 \ldots N+1$ **do**
  $\texttt{newmin} \leftarrow |b[i]|\, m + \max\left(s_p - b[i]t_p,\ s_m - b[i]t_m\right)$
  if $\texttt{newmin} < \texttt{min}$ **then**
    $\texttt{min} \leftarrow \texttt{newmin}$
    if $\texttt{ind}[i] \neq \texttt{none}$ **then**
      $t_p \leftarrow t_p - \mathbf{v}_{\texttt{ind}[i]} \left| \boldsymbol{\theta}_{\texttt{ind}[i]} \right|$
      $t_m \leftarrow t_m - \mathbf{v}_{\texttt{ind}[i]} \left| \boldsymbol{\theta}_{\texttt{ind}[i]} \right|$
      if $\boldsymbol{\theta}_{\texttt{ind}[i]} > 0$ **then**
        $s_p \leftarrow s_p - \mathbf{v}_{\texttt{ind}[i]} \boldsymbol{\theta}'_{\texttt{ind}[i]}$
        $s_m \leftarrow s_m - \mathbf{v}_{\texttt{ind}[i]} \boldsymbol{\theta}'_{\texttt{ind}[i]}$
      else
        $s_p \leftarrow s_p + \mathbf{v}_{\texttt{ind}[i]} \boldsymbol{\theta}'_{\texttt{ind}[i]}$
        $s_m \leftarrow s_m + \mathbf{v}_{\texttt{ind}[i]} \boldsymbol{\theta}'_{\texttt{ind}[i]}$
  else
    Check if there exists $\bar{\alpha} \in [b[i-1], b[i]]$ s.t. $\gamma_p(\bar{\alpha}) = \gamma_m(\bar{\alpha})$
    Return min($\texttt{newmin}$, $\eta(\alpha^{intersection})$)

and provide an upper bound on inner products with the query $\mathbf{X}_j \odot \boldsymbol{\theta}$ as follows:

$$\begin{aligned}\forall k \in [\![p]\!], \quad (\mathbf{X}_j \odot \boldsymbol{\theta})^\top \mathbf{X}_k &= \sum_{m \leq i} \mathbf{X}_{mj}\boldsymbol{\theta}_m \mathbf{X}_{mk} + \sum_{m > i} \mathbf{X}_{mj}\boldsymbol{\theta}_m \mathbf{X}_{mk} \\ &\leq \sum_{m \leq i} \mathbf{X}_{mj}\boldsymbol{\theta}_m \mathbf{X}_{mk} + \sum_{m > i;\ \boldsymbol{\theta}_m > 0} \mathbf{X}_{mj}\boldsymbol{\theta}_m \\ &= \sum_{m \leq i} \mathbf{X}_{mj}\boldsymbol{\theta}_m \mathbf{X}_{mk} + \boldsymbol{r}_i^+\end{aligned}$$

The bound involving $\mathbf{r}^-$ can be obtained analogously. These bounds simply assume there is a probe vector which has ones in front of every positive entry of the query and none in front of its negative entries, or the reverse. Once these bounds have been precomputed, the inner product between the query and a probe is computed up to a certain dimension, and every n_c dimensions we check whether there is a possibility that the inner product being computed becomes larger than the current maximum, or larger than λ. If it is impossible, then the probe can be safely discarded and the algorithm proceeds with the next probe. If not, the inner product is computed on n_c more dimensions and a new check is performed. For all our simulations and real data experiments, we set n_c to a default of 20. If a probe cannot be discarded then the algorithm updates when appropriate the active set $\mathcal{W}$ and/or the current maximum absolute inner product obtained $\mathbf{m}_j^{ref}$. For the pruning to be effective, we reorder the dimensions $1 \dots n$ so that queries are sorted in decreasing order in absolute value. As a consequence, the partial inner product bounds $\mathbf{r}_i^+$ and $\mathbf{r}_i^-$ are computed with the $n - i$ smallest entries in absolute value of the queries which makes them tighter than with any other ordering of the dimensions.

Algorithm S2 MIPS1

Input: $\mathbf{X} \in [0,1]^{n \times p}$, $\boldsymbol{\theta} \in \mathbb{R}^n$, $\mathcal{Q} \subset [\![p]\!]$, $\lambda \in \mathbb{R}$, $\mathcal{W} \subset [\![D]\!]$
Param: $n_c \in \mathbb{N}$
Output: $\mathcal{W}$, $\mathbf{m}^{ref}$.

Reorder the dimensions $1 \dots n$ such that $\boldsymbol{\theta}$ is sorted in descending order in absolute value and reorder the dimensions of $\mathbf{X}$ accordingly.
Reorder the columns of $\mathbf{X}$ in descending order of vector size.
for $j \in \mathcal{Q}$ **do** $\mathbf{m}_j^{ref} \leftarrow 0$
for $j \in \mathcal{Q}$ **do**
  Compute $\mathbf{r}^+ \in \mathbb{R}^n$ and $\mathbf{r}^- \in \mathbb{R}^n$ via (5) and (6).
  for $k \in [\![p]\!]$ **do**
    if $k \in \mathcal{Q}$ and $k > j$ then `continue`
    $d \leftarrow 0$ (inner product initialization); c $= 0$ (counter initialization);
    for $i \in \text{supp}(\boldsymbol{X}_j)$ **do**
      $d \leftarrow d + \mathbf{X}_{ij}\mathbf{X}_{ik}\boldsymbol{\theta}_i$
      $c \leftarrow c + 1$.
      if c `mod` $n_c = 0$ **then**
        if $\left|\,(d + \mathbf{r}_i^+)\,\right| < \min(\mathbf{m}_j^{ref}, \lambda)$ `and` $\left|(d + \mathbf{r}_i^-)\right| < \min(\mathbf{m}_j^{ref}, \lambda)$ **then** go to next probe.
    if $\mathbf{m}_j^{ref} < |\,d\,| < \lambda$ **then** set $\mathbf{m}_j^{ref} = |\,d\,|$
    if $|\,d\,| \geq \lambda$ and $\tau(k,j) \notin \mathcal{W}$ **then** add $\tau(k,j)$ to $\mathcal{W}$
return $\mathcal{W}, \boldsymbol{m}^{ref}$

We now compare $MIPS1$ to its naive counterpart (which we will call $Naive$ from now on)

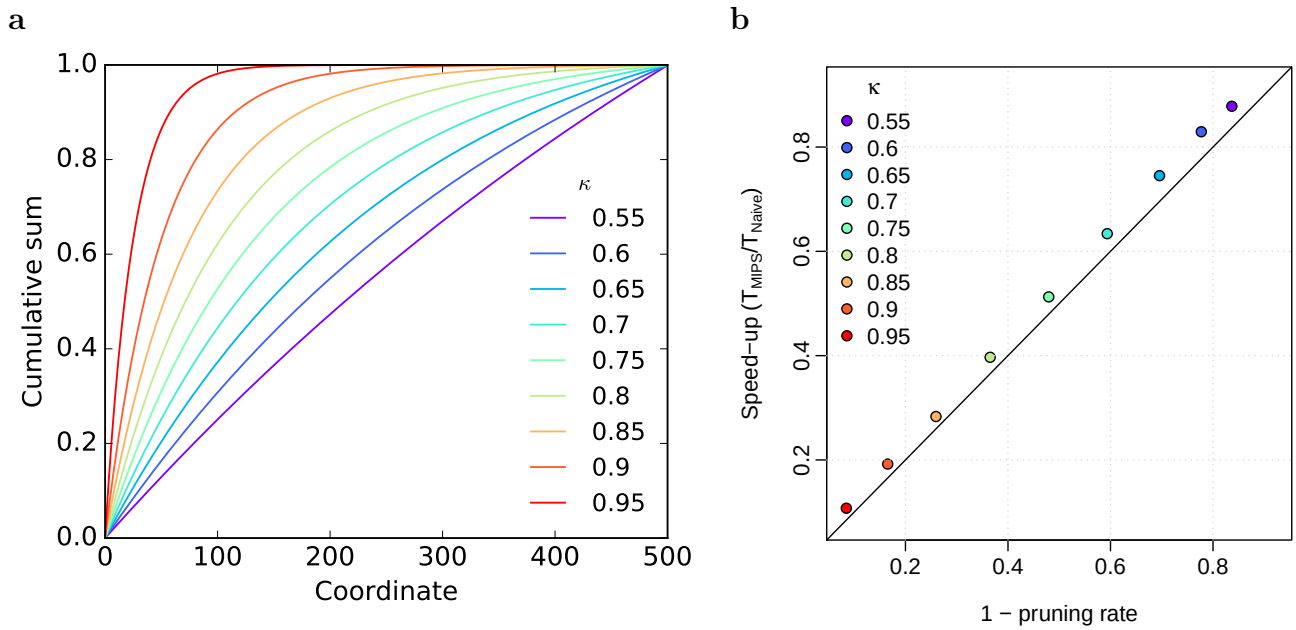

Figure S1 – Performances of MIPS1 on simulated data. (**a**) Cumulative sum of the vector obtained by sorting the positive entries of $\boldsymbol{\theta}_\kappa$ in decreasing order. (**b**) Speed-up obtained with $MIPS1$ compared to $Naive$ for different vectors $\boldsymbol{\theta}_\kappa$ as a function of the pruning rate. The pruning rate is defined as the average proportion of coordinates in the queries which are pruned.

on several benchmark datasets in order to assess the speed-up obtained with the pruning. To be more specific, $Naive$ is implemented similarly to $MIPS1$ except the lines specific to pruning, i.e., lines 5, 12 and 13 in Algorithm S2, are removed. The benchmark datasets we use are designed in such a way that the pruning rate achievable varies. To do this, we simulate a matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$, with $n = p = 1000$, where the features are drawn from a Bernoulli distribution, whose parameter is itself drawn from a uniform distribution $\mathcal{U}_{[0.1,0.5]}$. Then $\boldsymbol{\theta} \in \mathbb{R}^n$ is built in such a way that the cumulative sum of the vectors obtained by sorting $\boldsymbol{\theta}_{|\boldsymbol{\theta} \geq 0}$ and $|\boldsymbol{\theta}_{|\boldsymbol{\theta} < 0}|$ follows the function $f(x) = \frac{1}{1-e^{-\mu}}\left(1 - e^{-\mu x}\right), x \in \{0, 1\}$ for a given parameter $\mu \in \mathbb{R}^+$. The area under this cumulative sum, which is $\kappa(\mu) = \frac{1}{1-e^{-\mu}} - \frac{1}{\mu} \in [0.5, 1]$, characterises the different vectors $\boldsymbol{\theta}_\kappa$ obtained with different values of μ. Figure S1a shows how the cumulative sums are modified with μ. The interest of simulating different $\boldsymbol{\theta}_\kappa$ is that the rate of pruning achievable increases with κ: the closer κ is to 1, the higher the pruning rate. In the experiments presented hereafter, all p features were taken as queries, i.e., $\mathcal{Q} = [\![p]\!]$, and we took $\lambda = +\infty$ and $\mathcal{W} = \emptyset$. The results are presented in Figure S1b. The pruning rate, which we define as the average number of non-zero coordinates of the queries which were pruned out of their total number of non-zero coordinates, widely varies from 8% for $\kappa = 0.55$ to 84% for $\kappa = 0.95$. Moreover, the speed-up obtained with $MIPS1$ compared to $Naive$ is almost equal to 1 minus the pruning rate. That means $MIPS1$ is twice as fast as $Naive$ when it can prune half of the total number of coordinates.

We now compare the performance of $Naive$, $MIPS1$ and IL on the benchmark datasets (Figure S2). $MIPS1$ is the only method whose speed depends on κ since it is the only method to implement pruning. It has the same performance in terms of speed as $Naive$ for the lowest pruning rate, while it is as fast as IL for the highest pruning rates. For vectors $\boldsymbol{\theta}$ following classical distributions such as the gaussian distribution, $\kappa \approx 0.7$ and $MIPS1$ is therefore expected to be $\times 1.6$ times faster than

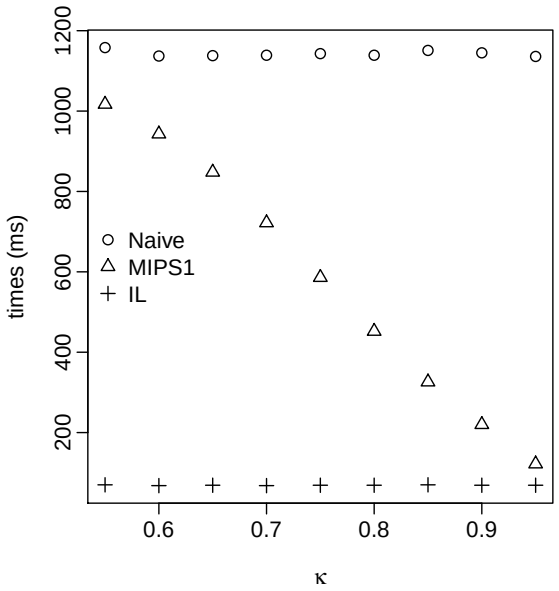

Figure S2 – Performance comparisons on simulated data. Time (in ms) taken by $Naive$, $MIPS1$ and IL to solve Maximum Inner Product Search problems with responses characterised by different κ.

$Naive$ but $\times 11$ times slower than IL. An analysis of the complexity of $MIPS1$ and IL can help to understand these results. For a given query, $MIPS1$ requires to compute inner products (although partially) with all p vectors in the database. In our implementation, the vectors are encoded as sparse vectors, i.e., the vector $\mathbf{X}_j$ is represented by the list of its non-zero indices. If we assume that the number of non-zero elements in the query is $|q|$ and that the total number of non-zero elements of the vectors in $\mathbf{X}$ in nnz, then $MIPS1$ has a $O(p|q| + nnz)$ complexity to compute the p inner products with the query. By contrast, the inverted index approach has a $O(|q|\frac{nnz}{n})$ complexity, where $\frac{nnz}{n}$ is the average length of an inverted index. As the number of non-zero elements $|q|$ in the query will typically be a fraction of the total number of samples n, the inverted index approach is expected to be faster than $MIPS1$ even though the pruning in $MIPS1$ can make it faster. This however may not be the case with dense data instead of sparse data.

D. SPP: depth-first vs breadth-first

The Safe Pattern Pruning algorithm presented in Nakagawa et al. (2016) deals with pairwise interactions but also higher-order interactions, and relies on a depth-first search scheme to explore the tree of patterns. However in our setting where we only consider pairwise interactions, we find that it is more efficient to implement a breadth-first search scheme for SPP. Indeed, the breadth-first search first identifies all the branches which can be screened. Then with this knowledge, we can restrict the number of interactions which are visited to those which only involve main effects whose corresponding branch was not screened. Basically, if we consider a case where p_s branches were screened among p branches, then the total number of nodes visited will be $p + \frac{(p-p_s)(p-p_s-1)}{2}$. Figure S3 illustrates the difference in performance obtained with the original SPP and the breadth-first search version in the case of pairwise interactions. The speed up obtained with the breadth-first search version ranges from $\times 1.2$ for $n = p = 1000$ to $\times 1.6$ for $n = 1000, p = 10000$. We therefore use the breadth-first search version of SPP as a comparison baseline in all our experiments.

a **b**

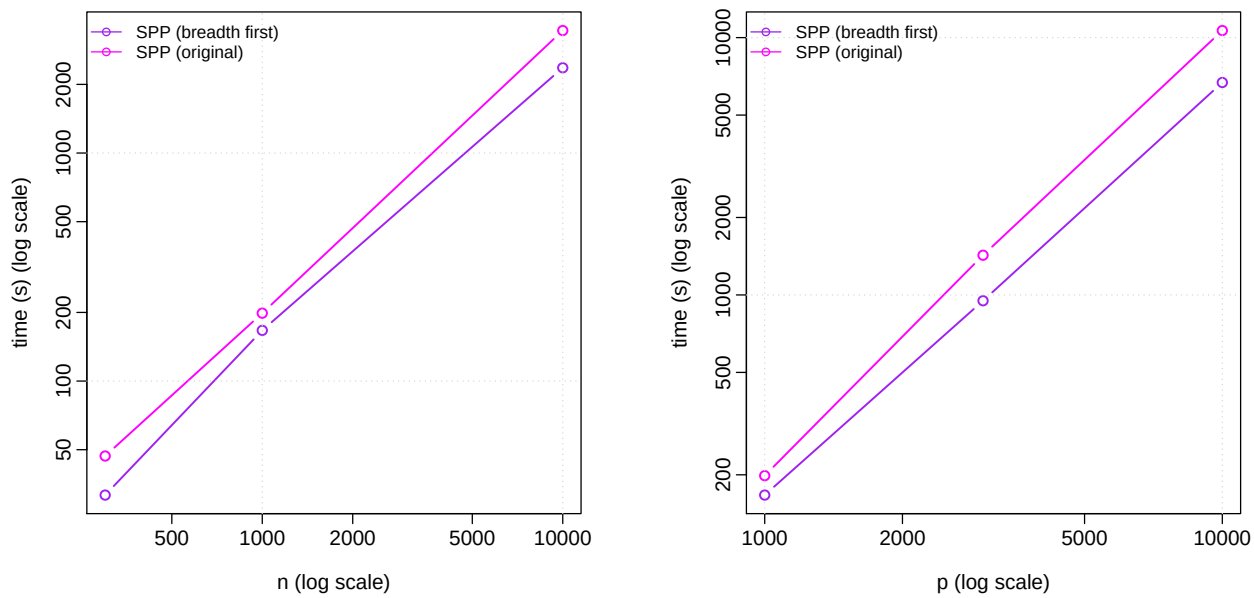

Figure S3 – **Safe Pattern Pruning performance on simulated data for an entire regularisation path**. The breadth-first search SPP (which is adapted to order-2 interactions only) is in purple and the original depth-first search SPP (which is adapted to order-2 interactions and more) is in magenta. (**a**) Time in seconds for $p = 1000$ fixed and n varied. (**b**) Time in seconds for $n = 1000$ fixed and p varied.

References

K. Nakagawa, S. Suzumura, M. Karasuyama, K. Tsuda, and I. Takeuchi. Safe Pattern Pruning: An Efficient Approach for Predictive Pattern Mining. In *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Min. - KDD '16*, pages 1785–1794, 2016.