Appendix

Faster Constraint Generation

A faster algorithm can be obtained by finding via binary search successively improved lower bounds on the cardinality k of nonzero entries in any optimal solution y^*. Only after we cannot further improve this lower bound we resort to a linear search on k, just as Algorithm 2 does. Below we describe the algorithm, and afterwards an explanatory proof of its correctness. The worst case complexity is still $O(V^2)$ since a linear search is required at the end, however in practice the algorithm is very fast since many if not most of the cardinalities k are never considered since they fall below the final lower bound.

Algorithm 3 Constraint Generation

```
1: Input: $(x, y)$, $\Psi$ $\theta$ Output: $y^*$ (index $n$ is left implicit)
2: Compute $I = \{i : y_i = 0\}$, and $z_{(0)}$, the subvector of $z$ restricted to $I$.
3: Compute the index set $(0^+)$ of the positive entries of $z_{(0)}$. Set $y^*_{(0^+)} = \mathbf{1}$.
4: $LB =$ cardinality of $(0^+)$
5: $k = LB$
6: Compute the index set $(V^+)$ of positive entries of $z = \Psi\theta - \frac{(1+\beta^2)y}{V+\beta^2\|y\|^2}$
7: $UB^+ =$ cardinality of $(V^+)$ (upper bound on no. positive entries of $z$)
8: repeat
9:     $z = \Psi\theta - \frac{(1+\beta^2)y}{k+\beta^2\|y\|^2}$
10:    Compute POS = $\#$ of positive entries in $z$
11:    if $k < POS$ then
12:        LB = POS
13:        $k = \lfloor (UB^+ + LB)/2 \rfloor$
14:    end if
15:    if $k > POS$ then
16:        $k = \lfloor (k + LB)/2 \rfloor$
17:    end if
18: until $k = LB$
19: (in the following for loop all computations can be restricted to the index
20: set $\mu = \{1, ..., V\} \backslash (0^+)$, since we know $y^*_{(0^+)} = \mathbf{1}$)
21: for $k = LB$ to V do
22:    $z = \Psi\theta - \frac{(1+\beta^2)y}{k+\beta^2\|y\|^2}$
23:    $y' = \operatorname{argmax}_{y \in \mathcal{Y}_k} \langle y, z \rangle$ (i.e. find top $k$ entries in $z$ in $O(V)$ time)
24:    CURRENT= $\max_{y \in \mathcal{Y}_k} \langle y, z \rangle$
25:    if CURRENT>MAX then
26:        MAX = CURRENT
27:        $y^* = y'$
28:    end if
29: end for
30: return $y^*$
```

Theorem 1 *Algorithm 3 finds an optimal solution to* $\operatorname{argmax}_{y \in \mathcal{Y}} \langle y, z_n \rangle$.

Proof There are two key ideas in the algorithm. The first is in lines 2-3. The subvector $z_{(0)}$ is constant, since it only depends on $(\Psi\theta)_{(0)}$. In particular, its own subvector obtained by restriction to the positive entries is also constant: $z_{(0^+)}$. The idea is that the cardinality of (0^+) is a lower bound on k, since removing a single 1 from $y^*_{(0^+)} = \mathbf{1}$ has two effects: (i) it will necessarily decrease the inner product $\left\langle y^*_{(0^+)}, z_{(0^+)} \right\rangle$ since all entries of $z_{(0^+)}$ are positive, and (ii) the remaining terms of the inner product $\langle y^*, z \rangle$ will either be decreased or kept constant. Therefore collectively the inner product $\langle y, z \rangle$ is decreased. This gives us a first lower bound. A succession of larger lower bounds can be obtained by incorporating a second idea, implemented in lines 6-18. We start with k as the first lower bound, and compute the amount of positive entries POS in the resulting z evaluated at that particular k (lines 9-10). The key insight now is that, if $POS > k$, then certainly POS is also a lower bound. This is true because by going from k to POS we necessarily *increase* the entries in z, so the indices that were positive continue to be positive and by the same previous argument any $k < POS$ will decrease the inner product. We then propose a new k halfway between the new lower bound and a (previously computed in line 7) upper bound UB^+ on the *number of positive entries of any* z. This is done because we cannot expect that $POS > k$ if $k = UB^+$, so the test in line 11 (which gives us a better lower bound) will never be accepted for $k > UB^+$. If however $POS < k$, then we don't have a new lower bound, and we decrease the proposed k halfway towards the current lower bound. The end of the binary search happens when the number of positive entries in z agrees with the proposed k, ie when the lower bound cannot be further improved. The fact that this will necessarily happen after some point follows from the fact that the sequence of lower bounds is increasing and UB^+ is an upper bound on this sequence. Once the binary search has finished, we simply revert to the naïve version of the algorithm, as described in Algorithm 2, and search for k between this best lower bound and V. ■