Assignment 2

(Large Scale Optimization)

Andreas Skiby Andersen, 202107332
Andreas Hyldegaard Hansen, 202106123
Aarhus Universitet

May 7, 2025

Contents

1 Task 1

We are given the subset selection problem with the purpose of finding at most k features such that the sum of squared residuals (L2-norm) of the OLS regression is minimized.

$$\begin{aligned} OLS(k) = \min ||Y - X\beta||_2^2 \\ \text{s.t.: } \sum_{j=1}^{p} z_j \leq k \\ M_j^- z_j \leq \beta_j \\ \beta_j \leq M_j^+ z_j \\ z_j \in \{0,1\} \end{aligned} \tag{1}$$

Let p be the total number of features, then $k \leq p$.
Let F be the indices of features $F = \{1,...,p\}$, let S be the set of indices for the selected features $S = \{1,...,k\} \subseteq F$ and let U be the set of indices for the unselected features such that $S \cup U = F$ and $S \cap U = \emptyset$.
We have the restricted OLS formula for β:

$$\hat{\beta}_j = \begin{cases} \left(\left(X_{(k)}^T X_{(k)} \right)^{-1} X_{(k)}^T Y \right)_j & j \in S \\ 0 & j \in U \end{cases} \tag{2}$$

Where $X_{(k)} = [:, S] \in \mathbb{R}^{n \times k}$ is the subset of columns corresponding to selected features with indices in S. We can see that for $j \in S$ we perform standard OLS and for $j \in U$ the coefficient is forced to be 0 such that we only have k non-zero coefficients.

We have used the metaheuristic, Iterated Local Search (ILS) to heuristically solve (1) and thereby obtain an upper bound on the cost $OLS(k)$. We have chosen ILS, since it is easy to implement, but allows for flexibility in the choice of perturbation, local search and acceptance criterion.

Our implementation is based upon numpy where vectors, matrices, matrix multiplications etc. are computed efficiently in C and Fortran.

1.1 Algorithm

Let $x^{(I)}$ be the initial solution, $x^{(C)}$ be the current solution, $x^{(P)}$ be the perturbation of $x^{(C)}$, $x^{(LS)}$ the local search of $x^{(P)}$ and let x^* the best solution found.
Note that if the initial solution has a cost that is smaller than the smallest float precision $\epsilon_{machine}$, then the algorithm immediately terminates with a cost of 0, as the solution is perfectly overfitted within the numeric precision we have available, meaning $f(x^{(I)}) \leq \epsilon_{machine}$.

```
1:  procedure ILS(X, Y, k, $x^{(I)}$, $I^{ILS}$, $I^{LS}$, d, accept)
2:      if $f(x^{(I)}) \leq \epsilon_{machine}$ then        ▷ Stop if initial solution is numerically close to 0
3:          Set $f(x^{(I)}) = 0$
4:          return $x^{(I)}$
5:      end if
6:
7:      $x^{(C)}$ = LocalSearch($X, Y, I^{LS}$, $x^{(I)}$, accept)
8:      $x^* = x^{(C)}$
9:      $i^{(u)} = 0$
10:
11:     for $i = 1, 2, ..., I^{ILS}$ do
12:         $S$ = get features from $x^{(C)}$
13:         $S'$ = Perturb($F, S, d$)
14:         $x^{(P)}$ = ComputeBeta($X, Y, S'$)
15:         $x^{(LS)}$ = LocalSearch($X, Y, F, I^{(LS)}, x^{(P)}$, accept)
16:
17:         if $f(x^{(LS)}) < f(x^*)$ then
18:             $x^* = x^{(LS)}$                                ▷ Update best solution
19:             $i^{(u)} = i$                                   ▷ Reset counter
20:             $d = 2$                          ▷ Decrease degree to intensify search
21:         end if
22:         $x^{(C)}, d, i^{(u)}$ = AcceptanceCriterion($x^{(C)}, x^{(LS)}, d, i, i^{(u)}$)
23:     end for
24:     return $x^*$
25: end procedure
```

```
1:  procedure ComputeBeta($X, y, S$)
2:      $X_{(k)}$ = X[:, S]                                    ▷ Get selected features
3:      $\beta_{(k)} = \left(X_{(k)}^T X_{(k)}\right)^{-1} X_{(k)}^T Y$
4:      $\beta = [0, ..., 0]$                                  ▷ Initialize all $\beta$'s as 0
5:      $\beta[S] = \beta_k$                     ▷ Set $\beta$ value for selected features
6:      $z = (Y - X\beta)^T (Y - X\beta)$
7:      x = $(\beta, z, S)$
8:      return x
9:  end procedure
```

1.2 Perturbation

Given a solution x, we perturb x to obtain a new random solution $x^{(P)}$ with degree d. Degree d defines the d-neighborhood of x where d number of features are swapped out, denoted as $x^{(P)} \in N_d(x)$.
First we sample a set of selected feature indices $S' \subseteq S$ of size d.
Then we sample a set of unselected feature indices $U' \subseteq U$ of size d such that $|S'| = |U'|$.
Then we swap the feature indices as $S = S \cup U' \setminus S'$ and $U = U \cup S' \setminus U'$.

```
procedure PERTURB($F$,$S$,$d$)
    $U \leftarrow F \setminus S$                          ▷ Compute unselected features

    $S' \leftarrow Sample(S,d)$
    $U' \leftarrow Sample(U,d)$
    $S = S \cup U' \setminus S'$                          ▷ Swap features
    return S
end procedure
```

1.3 Local search

Given a solution x, we local search the 1-neighborhood of x with the purpose of finding a local minimum.
We local search by swapping the pair of features (s^*, u^*) that results in the largest cost reduction where $s^* \in S$ and $u^* \in U$. The process is repeated for I^{LS} iterations or until there does not exist a pair of features that can reduce costs when swapped, hence a local minimum is found.
Our implementation supports best accept where all pairs (s^*, u^*) are evaluated and the pair with the largest cost reduction is chosen. It also supports first accept where the inner loop terminates after finding the first pair (s^*, u^*) that improves the cost.

```
procedure LocalSearch($X, Y, F, k, I^{LS}, x^{(I)}$, accept)
    $x^* = x^{(I)}$
    $S$ = get features from $x^*$
    $U = F \setminus S$                                   ▷ Compute unselected features
    for $i = 1, 2, ..., I^{LS}$ do
        $a^*$ = None                  ▷ Initialize variables for storing feature to add
        $r^*$ = None                                           ▷ and feature to remove
        for $s \in S$ do
            for $u \in U$ do
                $S' = S \cup u \setminus s$                          ▷ Swap features
                $x' = ComputeBeta(X, Y, S')$
                if $f(x') < f(x^*)$ then
                    $x^* = x'$                                  ▷ Update best solution
                    $a^* = u$                                   ▷ Update feature to add
                    $r^* = s$                                ▷ Update feature to remove
                    if accept=$'First'$ then
                        break out of inner loops   ▷ Stop when better solution is found
                    end if                               ▷ when using 'first accept'
                end if
            end for
        end for

        if $a^* \neq None \wedge r^* \neq None$ then
            $S = S \cup a^* \setminus r^*$                       ▷ Make the best update
        else
            break out of outer loop                                     ▷ Stop if done
        end if
    end for
    return $x^*$
end procedure
```

1.4 Initial solution

We can easily construct an initial solution by random sampling of k feature indices without replacement as S and perform OLS as (2).
But a good initialization may improve the quality of the obtained solution, so we have tried to make a greedy construction heuristic which utilizes the information contained in the integer-relaxation of (1).

$$\begin{aligned}
&\min ||Y - X\beta||_2^2 \\
&\text{s.t.:} \sum_{j=1}^{p} z_j \leq k \\
&M_j^- z_j \leq \beta_j \\
&\beta_j \leq M_j^+ z_j \\
&z_j \in [0,1]
\end{aligned} \tag{3}$$

Where we set $\forall j \in \{1,...,p\} : -M_j^- = M_j^+ = 100$.
Then we sort the indicators $sort(z_1, z_p)$ in decreasing order and select the k first indices as S and perform OLS as (2).

1.5 Acceptance Criterion

We want to accept the new solution $x^{(LS)}$ such that we avoid over-intensification of the search by only accepting better solutions, as this may result in a too narrow exploration of the search space.

We accept the new solution $x^{(LS)}$ if the cost is smaller than the cost of the current solution $x^{(C)}$, by $f(x^{(LS)}) < f(x^{(C)})$. If no solution has been accepted for 5 iterations, then $x^{(LS)}$ is accepted and the search is diversified by increasing the perturbation degree $d = min(k, d+1)$.

We keep track of the best found solution x^* and update it if $f(x^{(LS)}) < f(x^*)$. When it is updated, then the perturbation degree is reset to $d = 2$ to intensify the search.
The reason why perturbation degree is reset to $d = 2$ is that if $d = 1$ then no new local minima can be found in most cases. This is explained by the fact that if a pair of features are swapped, then in most cases the local search will redo the perturbation by swapping the pair back and thereby get trapped in the same local minimum.

Note that if $x^{(LS)}$ is the new best solution then $f(x^{(LS)}) < f(x^*) \leq f(x^{(C)})$ implies that we update the best solution $x^* = x^{(LS)}$, reset the degree $d = 2$ and update the current solution $x^{(C)} = x^{(LS)}$. Now the search continues from here, but focuses on intensification as the degree is reset to $d = 2$ and gradually expands to diversify the search if needed.

```
procedure AcceptanceCriterion($x^{(C)}, x^{(LS)}, d, i, i^{(u)}$)
    if $f(x^{(LS)}) < f(x^{(C)}) \vee i - i^{(u)} \geq 5$ then
        $x^{(C)} = x^{(LS)}$                          ▷ Move current solution
        if $i - i^{(u)} \geq 5$ then
            $d = \min(k, d+1)$                        ▷ Increase degree to diversify search
            $i^{(u)} = i$                             ▷ Reset counter
        end if
    end if
end procedure
```

2 Task 2

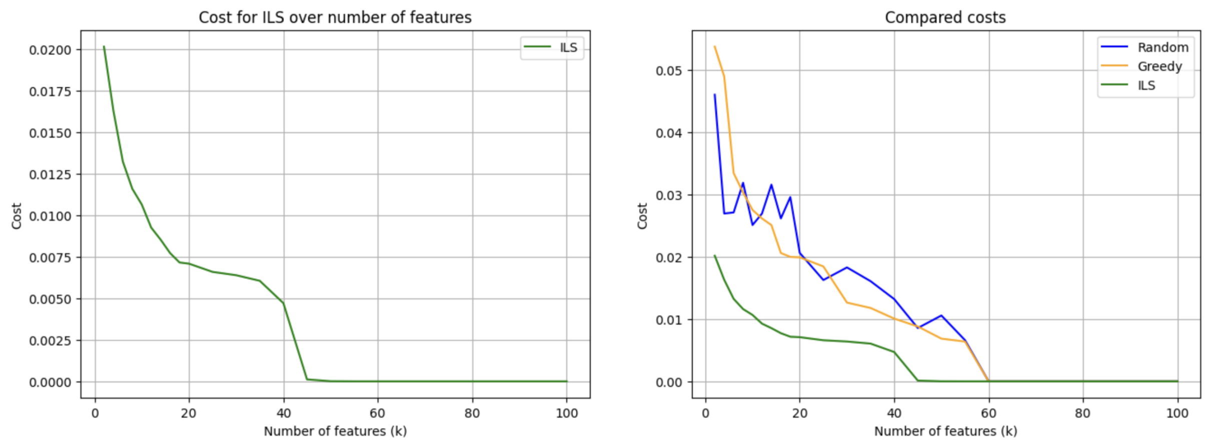

Figure 1: Cost OLS(k) for increasing number of features k (left) and comparison with a randomly sampled solution and greedy constructed solution (right).

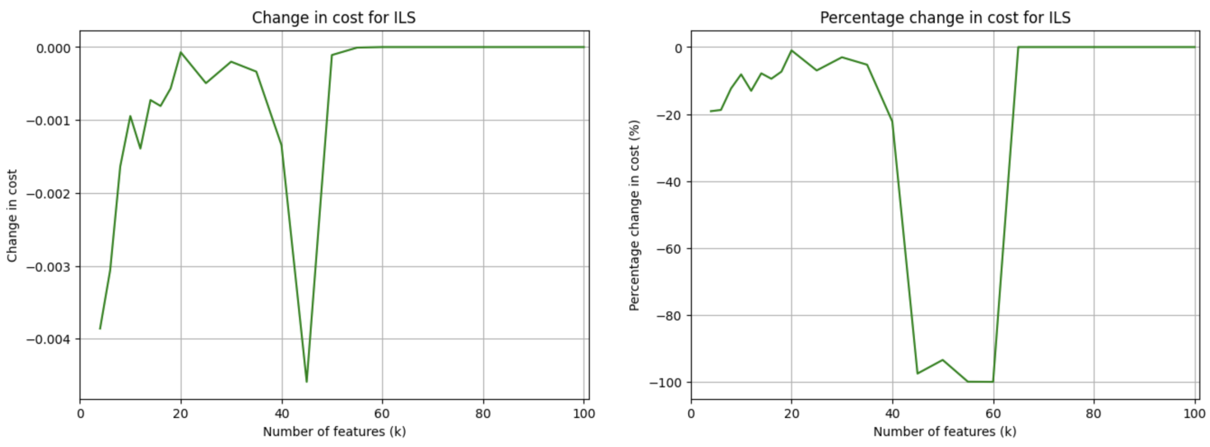

Figure 2: Absolute change in cost OLS(k) compared to OLS(k-1) for increasing number of features k (left) and relative change in cost (right).

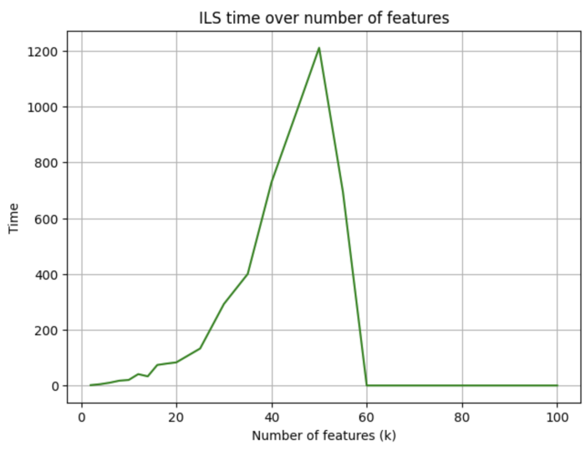

Figure 3: Time in seconds for running ILS over increasing number of features k.

3 Task 3

We have tested ILS for $k = [2,4,...,20,25,30,...,100]$, after testing 20 features we only test every 5th value of k due to the increased time complexity.
We always use best accept, but we could also have used first accept for faster convergence of local search, but our experimentation showed that it was only marginally faster.

Figure 1 shows the cost for ILS for increasing number of features k and compares it to a random solution and greedy solution. We see that the cost rapidly decreases from $k \in [2,20]$, but plateaus/slows down at $k \in [20,35]$. Then it starts overfitting at $k \in [40,45]$, until perfectly overfitting at $k \in [45,100]$. The change in cost is shown in figure 2, and it shows the same trend. First, the improvement goes towards 0, then plateaus and finally rapidly improves when overfitting.
Figure 1 also shows that the greedy solution is in most cases as bad as a randomly sampled solution, but our ILS procedure consistently improves the quality of the solution. We are surprised that the greedy solution is as bad as a random solution, as we expected that if $z_j \approx 1$ in the relaxation, then it would also be important in a feasible solution.

Figure 3 shows some order of polynomial increase in time when increasing the number of features k. We observe that the computation time starts to drop dramatically down to zero around $k = 50$, which makes sense, as this is also when we start to see overfitting in the plot of the cost. Hence, at this number of features, we cannot further improve on the initial solution, so the algorithm does not run.

The polynomial increasing computation time, is what we expected to see, as we loop over the number of features multiple times when performing the local search.