A Detailed algorithm description and proof of Theorem 1

Algorithm 4 Compute the pseudoinverse of $\sum_{i=1}^k x_i x_i^\intercal$

procedure PseudoInv($x_1, \dots, x_k$)
  $c_1, \dots, c_k, u_1, \dots, u_k \leftarrow$ Gram-Schmidt($x_1, \dots, x_k$) ▷ $x_i = \sum_{j=1}^k c_{ij} u_j$.
  $C \leftarrow \sum_{i=1}^k c_i c_i^\intercal$
  $\lambda_1, a_1, \dots, \lambda_k, a_k \leftarrow$ Eigendecompose(C) ▷ Eigenvalue λ_i has corresponding eignevector a_i
  for $i = 1, \dots, k$ **do**
    $v_i \leftarrow \sum_{j=1}^k a_{ij} u_j$
  end for
  return $\lambda_1^{-1}, v_1, \dots, \lambda_k^{-1}, v_k$
end procedure

Algorithm 5 Fast multiplication by pseudoinverse

procedure FastMult($S^{-1}, \nabla L$) ▷ S^{-1} must be given in its low-rank form $S^{-1} = \sum_{i=1}^k \lambda_i^{-1} v_i v_i^\intercal$
  return $\sum_{i=1}^k (\lambda_i^{-1}(v_i^\intercal \nabla L)) v_i$ ▷ Compute according to the specified parenthesization
end procedure

We take a gradient step in the direction specified by the synthetic LKO points $(x_i, \hat{y}_i^{\setminus k})$, $i = 1, \dots, k$. That is, we update

$$\theta^{\text{res}} = \theta^{\text{full}} - \alpha \sum_{i=1}^k (\theta^{\text{full}\intercal} x_i - \hat{y}_i^{\setminus k}) x_i. \tag{3}$$

Ordinarily, the parameter α is a scalar which specifies the step size. For our purposes, we will replace α with a "step matrix" A.

Proof of Theorem 1. Recalling that $\hat{y}_i^{\setminus k} = \theta^{\setminus k\intercal} x_i$, we can rewrite equation (3):

$$\begin{aligned}\theta^{\text{full}} - A \sum_{i=1}^k (\theta^{\text{full}\intercal} x_i - \hat{y}_i^{\setminus k}) x_i &= \theta^{\text{full}} - A \sum_{i=1}^k (\theta^{\text{full}\intercal} x_i - \theta^{\setminus k\intercal} x_i) x_i \\ &= \theta^{\text{full}} - A \left(\sum_{i=1}^k x_i x_i^\intercal \right) (\theta^{\text{full}} - \theta^{\setminus k}). \end{aligned} \tag{4}$$

Let $B = \sum_{i=1}^k x_i x_i^\intercal$. Note that $\text{range}(B) = \text{span}\{x_1, \dots, x_k\} \triangleq V_k$. Due to the form that B has, we can efficiently compute its eigendecomposition $B = V \Lambda V^\intercal$, where $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_k, 0, \dots, 0)$, $\lambda_1, \dots, \lambda_k$ are the nonzero eigenvalues of B, and $V V^\intercal = I$. We then define

$$A = V \Lambda^\dagger V^\intercal, \qquad \Lambda^\dagger = \text{diag}(\lambda_1^{-1}, \dots, \lambda_k^{-1}, 0, \dots, 0). \tag{5}$$

This choice of A gives us $AB = \sum_{i=1}^k v_i v_i^\intercal = \text{proj}_{V_k}$, and therefore the update (4) is equivalent to

$$\theta^{\text{full}} + \text{proj}_{V_k}(\theta^{\setminus k} - \theta^{\text{full}}). \tag{6}$$

This establishes the first claim in Theorem 1. It remains to perform the computational cost calculation. We analyze the computational cost of the algorithm by breaking it down into several submodules.

Step 1: Computing $\hat{y}_i^{\setminus k}$, $i = 1, \dots, k$

By Theorem 4, this step can be accomplished in $O(k^3)$ time.

Step 2: Finding the eigendecomposition of $\sum_{i=1}^k x_i x_i^\intercal$

We will show that this step can be completed in $O(k^2 d)$ time. We compute the eigendecomposition of $B \equiv \sum_{i=1}^k x_i x_i^\intercal$ as follows.

I. Perform Gram-Schmidt on $x_1, \ldots, x_k$ to recover $u_1, \ldots, u_k$ and coefficients c_{ij}. computational cost: $O(k^2 d)$.

(a) In the i-th step, we set $w_i = x_i - ((x_i^\intercal u_1)u_1 + \cdots + (x_i^\intercal u_{i-1})u_{i-1})$, followed by $u_i = w_i / \|w_i\|$. Naively computing the dot products, scalar-vector products, and vector sums for step i takes $O(id)$ time. Summing over the steps, the total time to perform Gram-Schmidt is $\sum_{i=1}^k O(id) = O(k^2 d)$.

(b) From the i-th step of Gram-Schmidt, we see that

$$x_i = (x_i^\intercal u_1)u_1 + \cdots + (x_i^\intercal u_{i-1})u_{i-1} + \|w_i\| u_i$$
$$\therefore c_{ij} = \begin{cases} x_i^\intercal u_j, & 1 \leq j < i \\ \|w_i\|, & j = i \\ 0, & j > i \end{cases}$$

We can store these coefficients as we compute them during the Gram-Schmidt procedure without increasing the asymptotic time complexity of this step.

II. Eigendecompose the $k \times k$ matrix $C = \sum_{i=1}^k c_i c_i^\intercal$ and recover the eigendecomposition of B. computational cost: $O(k^2 d)$.

(a) We claim that the first k eigenvalues of B are identical to the eigenvalues of C, and that the eigenvectors of B can easily be recovered from the eigenvectors of C. In particular, if $a_1, \ldots, a_k \in \mathbb{R}^k$ are the eigenvectors of C, then $v_i = a_{i1}u_1 + \ldots + a_{ik}u_k$ is the i-th eigenvector of B.
To see this, note that $R(B) = \text{span}\{x_1, \ldots, x_k\}$, so any eigenvector for a nonzero eigenvalue of B must be in the span of the x_i. Since $u_1, \ldots, u_k$ have the same span as the x_i, if v is an eigenvector for B with nonzero eigenvalue λ, we can write $v = b_1 u_1 + \cdots + b_k u_k$. Let $b = (b_1, \ldots, b_k)^\intercal \in \mathbb{R}^k$. We can also rewrite $B = \sum_{i=1}^k x_i x_i^\intercal = \sum_{i,j,\ell=1}^k c_{ij} c_{i\ell} u_j u_\ell^\intercal$. Combining these facts yields

$$\begin{aligned} Bv &= \sum_{i,j,\ell=1}^k b_\ell c_{i\ell} c_{ij} u_j \\ &= \sum_{i,j=1}^k (c_i^\intercal b) c_{ij} u_j \\ &= \lambda b_1 u_1 + \cdots + \lambda b_k u_k. \end{aligned}$$

Since the u_js are linearly independent, we can equate coefficients. Doing so shows that $\lambda b_j = \sum_{i=1}^k (c_i^\intercal b) c_{ij}$ for all $j = 1, \ldots, k$. Vectorizing these equations, we have that

$$Cb = \sum_{i=1}^k c_i (c_i^\intercal b) = \lambda b.$$

This chain of equalities holds in reverse order as well, so we conclude that v is an eigenvector for B with nonzero eigenvalue λ iff b is an eigenvector for C with eigenvalue λ. Since we know that the remaining eigenvalues of B are 0, it suffices to find an eigendecomposition of C. Forming C takes $O(k^3)$ time, and finding its eigendecomposition can be done (approximately) in $O(k^3)$ time, see (Pan and Chen, 1999). Finally, converting each eigenvector a_i for C into an eigenvector for B takes $O(kd)$ time (we set $v_i = a_{i1}u_1 + \cdots + a_{ik}u_k$), so converting all k of them takes $O(k^2 d)$ time.

(b) Since we know B is rank k, the remaining eigenvalues are 0 and any orthonormal extension of the orthonormal eigenvectors $v_1, \ldots, v_k$ computed in step 2 will suffice to complete an orthonormal basis of eigenvectors for $\mathbb{R}^d$. Let $v_{k+1}, \ldots, v_d$ be any such extension. This gives us a complete orthonormal basis of eigenvectors $v_1, \ldots, v_d$ for $\mathbb{R}^d$ with associated eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_k > \lambda_{k+1} = \ldots = \lambda_d = 0$. We now define A as in equation (5). Observe that since $(\Lambda^\dagger)_{ii} = 0$ for $i > k$, we can compute A without needing to know the values of $v_{k+1}, \ldots, v_d$:

$$A = \sum_{i=1}^k \lambda_i^{-1} v_i v_i^\intercal. \tag{7}$$

Step 3: Performing the update

We will show that this step can be completed in $O(kd)$ time. Recall the form of the projective residual update:

$$\theta^{\text{res}} = \theta^{\text{full}} - A\sum_{i=1}^{k}(\theta^{\text{full}\intercal}x_i - \hat{y}_i')x_i.$$

I. Form the vector $\nabla L = \sum_{i=1}^{k}(\theta^{\text{full}\intercal}x_i - \hat{y}_i')x_i$. (This is the gradient of the loss on the synthetic datapoints.) computational cost: $O(kd)$.

II. Compute the step $A\nabla L$. computational cost: $O(kd)$.

(a) Rather than performing the computationally expensive operations of forming the matrix A, then doing a $d \times d$ matrix-vector multiplication, we use the special form of A. Namely, we have

$$\begin{aligned} A\nabla L &= \left(\sum_{i=1}^{k}\lambda_i^{-1}v_iv_i^\intercal\right)s \\ &= \sum_{i=1}^{k}(\lambda_i^{-1}(v_i^\intercal s))v_i. \end{aligned} \tag{8}$$

(b) Each term in the summand (8) can be computed in $O(d)$ time, so we can compute the entire sum in $O(kd)$ time.

III. Update $\theta^{\text{res}} = \theta^{\text{full}} - A\nabla L$. computational cost: $O(d)$.

Since we have assumed $k \leq d$, the total computational cost of the algorithm is therefore $O(k^3)+O(k^2d)+O(kd) = O(k^2d)$ as desired. □

Note that the crucial step of computing the exact leave-k-out predicted y-values may vary depending on the the specific instance of least squares we found ourselves in (e.g. with or without regularization or weighting, see Appendix E), but the rest of the algorithm remains exactly the same.

B Performance analysis for outlier removal

In this section we prove Theorem 5. We also quantify the behavior of the true step $\theta^{\text{full}} - \theta^{\setminus 1}$ as the outlier size λ grows.

Proposition 7. *Let D^{full} be as in Theorem 5. As $\lambda \to \infty$, $\theta^{\text{full}} - \theta^{\setminus 1} \to C\hat{\Sigma}^{-1}x_1$, where $\hat{\Sigma}$ is the empirical covariance matrix for the dataset $D^{\setminus 1}$ and C is a (data-dependent) scalar constant.*

Proof. Departing slightly from the notation in section 2, let X and Y denote the feature matrix and response vector, respectively, for the dataset $D^{\setminus 1}$. The exact values of θ^{full} and $\theta^{\setminus 1}$ are then given by

$$\begin{aligned} \theta^{\setminus 1} &= (X^\intercal X)^{-1}XY \\ \theta^{\text{full}} &= (X^\intercal X + \lambda^2 x_1 x_x^\intercal)^{-1}(XY + \lambda^2 y_1 x_1). \end{aligned}$$

We can expand the expression for θ^{full} with the Sherman-Morrison formula:

$$\theta^{\text{full}} = \left[(X^\intercal X)^{-1} - \frac{\lambda^2(X^\intercal X)^{-1}x_1x_1^\intercal(X^\intercal X)^{-1}}{1+\lambda^2 x_1^\intercal(X^\intercal X)^{-1}x_1}\right]\cdot(XY + \lambda^2 y_1 x_1). \tag{9}$$

From equation (9), we see that the actual step is

$$\theta^{\text{full}} - \theta^{\setminus 1} = \lambda^2 y_1 (X^\intercal X)^{-1} x_1 - \frac{\lambda^2 (X^\intercal X)^{-1} x_1 x_1^\intercal (X^\intercal X)^{-1}}{1 + \lambda^2 x_1^\intercal (X^\intercal X)^{-1} x_1} (XY + \lambda^2 y_1 x_1)$$

$$= (X^\intercal X)^{-1} \left(y_1 \underbrace{\lambda^2 \left[I - \frac{\lambda^2 x_1 x_1^\intercal (X^\intercal X)^{-1}}{1 + \lambda^2 x_1^\intercal (X^\intercal X)^{-1} x_1} \right] x_1}_{\text{(I)}} - \underbrace{\frac{\lambda^2 x_1 x_1^\intercal (X^\intercal X)^{-1}}{1 + \lambda^2 x_1^\intercal (X^\intercal X)^{-1} x_1} XY}_{\text{(II)}} \right). \tag{10}$$

Let us analyze the behavior of the terms (I) and (II) in equation (10) as $\lambda \to \infty$. Term (II) is straightforward: the λ^2 terms dominate both the numerator and the denominator, so we have

$$\text{(II)} \longrightarrow \frac{x_1 x_1^\intercal (X^\intercal X)^{-1} XY}{x_1^\intercal (X^\intercal X)^{-1} x_1} = \frac{x_1^\intercal (X^\intercal X)^{-1} XY}{x_1^\intercal (X^\intercal X)^{-1} x_1} x_1 \qquad \text{as } \lambda \to \infty.$$

The term (I) is slightly more delicate, since the first-order behavior of (I) without multiplication by λ^2 tends to 0; however, the multiplication by λ^2 means that this term does not vanish. Observing that (I) can be rewritten as

$$\lambda^2 \left[1 - \frac{\lambda^2 x_1^\intercal (X^\intercal X)^{-1} x_1}{1 + \lambda^2 x_1^\intercal (X^\intercal X)^{-1} x_1} \right] x_1,$$

we have reduced our analysis of (I) to determining the leading order behavior of a function of the form

$$f(\lambda) \equiv \lambda^2 \left[1 - \frac{c\lambda^2}{1 + c\lambda^2} \right]. \tag{11}$$

(In our case, $c = x_1^\intercal (X^\intercal X)^{-1} x_1$.) A Taylor expansion of (11) shows that $f(\lambda) = c^{-1} + O(\lambda^{-2})$, and thus we have

$$\text{(I)} \longrightarrow \frac{x_1}{x_1^\intercal (X^\intercal X)^{-1} x_1} \qquad \text{as } \lambda \to \infty.$$

Substituting the limits of (I) and (II) into equation (10), we see that

$$\theta^{\text{full}} - \theta^{\setminus 1} \to (X^\intercal X)^{-1} \left[\frac{y_1}{x_1^\intercal (X^\intercal X)^{-1} x_1} x_1 - \frac{x_1^\intercal (X^\intercal X)^{-1} XY}{x_1^\intercal (X^\intercal X)^{-1} x_1} x_1 \right]$$

$$= \underbrace{\frac{y_1 - x_1^\intercal (X^\intercal X)^{-1} XY}{x_1^\intercal (X^\intercal X)^{-1} x_1}}_{C'} (X^\intercal X)^{-1} x_1. \tag{12}$$

The result follows by multiplying and dividing (12) by a factor of n (so $C = C'/n$ and the other factor of n gets pulled into $(X^\intercal X)^{-1}$ to yield $\hat{\Sigma}^{-1}$). □

Proof of Theorem 5. We will analyze $\theta^{\text{inf}} - \theta^{\setminus 1}$ and show that the limit of this difference is the same as that of $\theta^{\text{full}} - \theta^{\setminus 1}$ as $\lambda \to \infty$; it immediately follows that $\theta^{\text{inf}} \to \theta^{\text{full}}$. By the exactness of the Newton update for linear regression, we have

$$\theta^{\setminus 1} = \theta^{\text{full}} + (X^\intercal X)^{-1} \lambda^2 (\theta^{\text{full}\intercal} x_1 - y_1) x_1.$$

By definition, the influence parameters are given by

$$\theta^{\text{inf}} = \theta^{\text{full}} + (X^\intercal X + \lambda^2 x_1 x_1^\intercal)^{-1} \lambda^2 (\theta^{\text{full}\intercal} x_1 - y_1) x_1.$$

Subtracting these two expressions yields

$$\theta^{\text{inf}} - \theta^{\setminus 1} = \lambda^2 (\theta^{\text{full}\intercal} x_1 - y_1) \cdot [(X^\intercal X + \lambda^2 x_1 x_1^\intercal)^{-1} - (X^\intercal X)^{-1}] x_1. \tag{13}$$

We analyze the terms in the RHS of (13) separately.

First, note that by the Sherman-Morrison formula, we have

$$[(X^\intercal X + \lambda^2 x_1 x_1^\intercal)^{-1} - (X^\intercal X)^{-1}]x_1 = \frac{-\lambda^2 (X^\intercal X)^{-1} x_1 x_1^\intercal (X^\intercal X)^{-1}}{1 + \lambda^2 x_1^\intercal (X^\intercal X)^{-1} x_1} x_1$$

$$= \frac{-\lambda^2 x_1^\intercal (X^\intercal X)^{-1} x_1}{1 + \lambda^2 x_1^\intercal (X^\intercal X)^{-1} x_1} (X^\intercal X)^{-1} x_1 \tag{14}$$

$$\to -(X^\intercal X)^{-1} x_1. \tag{15}$$

Equation (15) follows since the numerator and denominator of (14) have the same leading order behavior in λ.

Next, we analyze the term $\theta^{\text{full}\intercal} x_1 - y_1$. We begin by substituting the expression for θ^{full} and once more applying the Sherman-Morrison formula:

$$\begin{aligned} x_1^\intercal \theta^{\text{full}} - y_1 &= x_1^\intercal (X^\intercal X + \lambda^2 x_1 x_1^\intercal)^{-1} (XY + \lambda^2 x_1 y_1) - y_1 \\ &= x_1^\intercal \left[(X^\intercal X)^{-1} - \frac{\lambda^2 (X^\intercal X)^{-1} x_1 x_1^\intercal (X^\intercal X)^{-1}}{1 + \lambda^2 x_1^\intercal (X^\intercal X)^{-1} x_1} \right] (XY + \lambda^2 x_1 y_1) - y_1 \\ &= x_1^\intercal (X^\intercal X)^{-1} \left[\underbrace{\left(I - \frac{\lambda^2 x_1 x_1^\intercal (X^\intercal X)^{-1}}{1 + \lambda^2 x_1^\intercal (X^\intercal X)^{-1} x_1} \right) XY}_{(i)} + y_1 \underbrace{\left(\lambda^2 \left[I - \frac{\lambda^2 x_1 x_1^\intercal (X^\intercal X)^{-1}}{1 + \lambda^2 x_1^\intercal (X^\intercal X)^{-1} x_1} \right] x_1 \right)}_{(ii)} \right] - y_1 \end{aligned} \tag{16}$$

We rearrange (i) and (ii) and then Taylor expand:

$$\begin{aligned} (i) &= XY - \frac{\lambda^2 x_1^\intercal (X^\intercal X)^{-1} XY}{1 + \lambda^2 x_1^\intercal (X^\intercal X)^{-1} x_1} x_1 \\ &= XY - \frac{x_1^\intercal (X^\intercal X)^{-1} XY}{x_1^\intercal (X^\intercal X)^{-1} x_1} x_1 - \frac{x_1^\intercal (X^\intercal X)^{-1} XY}{(x_1^\intercal (X^\intercal X)^{-1} x_1)^2} \lambda^{-2} x_1 + O(\lambda^{-4}) \\ (ii) &= \lambda^2 \left[1 - \frac{\lambda^2 x_1^\intercal (X^\intercal X)^{-1} x_1}{1 + \lambda^2 x_1^\intercal (X^\intercal X)^{-1} x_1} \right] x_1 \\ &= \frac{x_1}{x_1^\intercal (X^\intercal X)^{-1} x_1} - \frac{\lambda^{-2} x_1}{(x_1^\intercal (X^\intercal X)^{-1} x_1)^2} + O(\lambda^{-4}) \end{aligned}$$

Substituting these equations into equation (16) yields

$$\begin{aligned} x_1^\intercal \theta^{\text{full}} - y_1 &= x_1^\intercal (X^\intercal X)^{-1} \Bigg[XY - \frac{x_1^\intercal (X^\intercal X)^{-1} XY}{x_1^\intercal (X^\intercal X)^{-1} x_1} x_1 + \frac{y_1 x_1}{x_1^\intercal (X^\intercal X)^{-1} x_1} \\ &\qquad - \frac{x_1^\intercal (X^\intercal X)^{-1} XY}{(x_1^\intercal (X^\intercal X)^{-1} x_1)^2} \lambda^{-2} x_1 - \frac{\lambda^{-2} y_1 x_1}{(x_1^\intercal (X^\intercal X)^{-1} x_1)^2} \Bigg] - y_1 + O(\lambda^{-4}) \\ &= \left(x_1^\intercal (X^\intercal X)^{-1} XY - \frac{x_1^\intercal (X^\intercal X)^{-1} XY}{x_1^\intercal (X^\intercal X)^{-1} x_1} x_1^\intercal (X^\intercal X)^{-1} x_1 + \frac{y_1 x_1^\intercal (X^\intercal X)^{-1} x_1}{x_1^\intercal (X^\intercal X)^{-1} x_1} - y_1 \right) \\ &\quad + \left(\frac{x_1^\intercal (X^\intercal X)^{-1} XY - y_1}{x_1^\intercal (X^\intercal X)^{-1} x_1} \right) \lambda^{-2} + O(\lambda^{-4}) \\ &= \frac{x_1^\intercal (X^\intercal X)^{-1} XY - y_1}{x_1^\intercal (X^\intercal X)^{-1} x_1} \lambda^{-2} + O(\lambda^{-4}). \end{aligned} \tag{17}$$

Finally, we substitute the expressions from equations (15) and (17) into (13) to obtain

$$\begin{aligned}\theta^{\text{inf}} - \theta^{\setminus 1} &= \left(-\frac{(X^\intercal X)^{-1}x_1 x_1^\intercal (X^\intercal X)^{-1}}{x_1^\intercal (X^\intercal X)^{-1} x_1} + O(\lambda^{-2})\right)\lambda^2 \left(\frac{x_1^\intercal (X^\intercal X)^{-1} XY - y_1}{x_1^\intercal (X^\intercal X)^{-1} x_1}\lambda^{-2} + O(\lambda^{-4})\right) x_1 \\ &= \frac{y_1 - x_1^\intercal (X^\intercal X)^{-1} XY}{x_1^\intercal (X^\intercal X)^{-1} x_1}(X^\intercal X)^{-1} x_1 + O(\lambda^{-2}).\end{aligned}$$

Note that this has the same limiting value as $\theta^{\text{full}} - \theta^{\setminus 1}$ as $\lambda \to \infty$ (see equation (12)) and we are done. □

C Proof of Theorem 4

We prove Theorem 4 for the case of ordinary least squares. We generalize this logic to weighted, ridge regularized least squares in Appendix D.

Proof of Theorem 4. We make use of the analytic form of the parameters for least squares linear regression. Given a dataset $\{(x_i, y_i)\}_{i=1}^n$, we have $\theta^{\text{full}} = (X^\intercal X)^{-1} XY$, with X, Y defined as in section 2. The predictions for the fitted model on the dataset are then given by

$$\hat{Y} = X\theta^{\text{full}} = \underbrace{(X(X^\intercal X)^{-1} X^\intercal)}_{H} Y, \tag{18}$$

where $H = X(X^\intercal X)^{-1} X^\intercal$ is the so-called hat matrix. As previously mentioned, we assume that we already have access to H after the model has been trained on the full dataset.

Next, observe that

$$\theta^{\setminus k} = \underset{\theta}{\text{argmin}} \left[\sum_{i=1}^{k} (\theta^\intercal x_i - \hat{y}_i^{\setminus k})^2 + \sum_{i=k+1}^{n} (\theta^\intercal x_i - y_i)^2\right]$$

since $\theta^{\setminus k}$ minimizes both sums individually. It follows from equation (18) that $HY' = \hat{Y}_{\setminus k}$, where $Y' = (\hat{y}_1^{\setminus k}, \ldots, \hat{y}_k^{\setminus k}, y_{k+1}, \ldots, y_n)^\intercal$ and $\hat{Y}_{\setminus k} = (\hat{y}_1^{\setminus k}, \ldots, \hat{y}_n^{\setminus k})^\intercal$.

This relation $HY' = \hat{Y}_{\setminus k}$ allows us to derive a system of linear equations between $\hat{y}_i^{\setminus k}$ for $i = 1, \ldots, k$. Namely, if we define $r_i = y_i - \hat{y}_i$, $r = (r_1, \ldots, r_k)^\intercal$, $r_i^{\setminus k} = y_i - \hat{y}_i^{\setminus k}$, and $r^{\setminus k} = (r_1^{\setminus k}, \ldots, r_k^{\setminus k})^\intercal$, we have

$$r_i^{\setminus k} = \frac{r_i + \sum_{j \neq i} h_{ij} r_j^{\setminus k}}{1 - h_{ii}}, \tag{19}$$

where h_{ij} are the entries of H. Vectorizing equation (19) and solving yields

$$r^{\setminus k} = (I - T)^{-1}(\tfrac{r_1}{1-h_{11}}, \ldots, \tfrac{r_k}{1-h_{kk}})^\intercal, \tag{20}$$

where $T_{ij} = \mathbf{1}\{i \neq j\}\frac{h_{ij}}{1-h_{jj}}$. Since this is a system of k linear equations in k unknowns, we can solve it in time $O(k^3)$ via simple Gaussian elimination. The values $\hat{y}_i^{\setminus k}$ can then be easily recovered in an additional $O(k)$ time by noting that $\hat{y}_i^{\setminus k} = y_i - r_i^{\setminus k}$. □

D Generalization of Theorem 4 to weighted, ridge regularized least squares

Refer to Appendix C. We can generalize our method for computing the predictions of the LKO model to weighted least squares with ridge regularization. Let $w \succeq 0 \in \mathbb{R}^n$ denote a (fixed) weight vector and $\lambda \geq 0$ be the regularization strength, which we require to be fixed independent of the number of samples. The weighted, regularized loss is given by

$$\begin{aligned}L^{\text{full}}(\theta) &= \frac{1}{2}\left(\sum_{i=1}^{n} w_i(\theta^\intercal x_i - y_i)^2 + \lambda\|\theta\|^2\right) \\ &= \frac{1}{2}[(X\theta - Y)^\intercal W(X\theta - Y) + \lambda\|\theta\|^2].\end{aligned}$$

The gradient is therefore

$$\nabla L^{\text{full}}(\theta) = X^\intercal W X\theta - X^\intercal W Y + \lambda\theta \tag{21}$$

Using equation (21), we see that $\nabla L^{\text{full}} = 0$ when

$$\theta^{\text{full}} = (X^\intercal W X + \lambda I)^{-1} X^\intercal W Y.$$

Predictions are therefore given by

$$X\theta^{\text{full}} = \underbrace{X(X^\intercal W X + \lambda I)^{-1} X^\intercal W}_{H_{\lambda,w}} Y.$$

If we replace H in equation (18) with $H_{\lambda,w}$, the same logic carries through. Note that the regularization strength needs to be fixed for us to use the same trick, i.e. to write

$$\theta^{\setminus k} = \operatorname*{argmin}_\theta \sum_{i=1}^{k} w_i(\theta^\intercal x_i - \hat{y}_i^{\setminus k})^2 + \sum_{i=k+1}^{n} w_i(\theta^\intercal x_i - y_i)^2 + \lambda\|\theta\|^2$$

with $\hat{y}_i^{\setminus k} = \theta^{\setminus k\intercal} x_i$ the predicted y-value for the LKO model. In this case, we can compute the LKO prediction values efficiently ($O(k^3)$ time when we precompute $H_{\lambda,w}$). Theorem 4 therefore holds in this more general setting as well.

E Proof of Theorem 6

Proof. For logistic regression, we use the loss function

$$L(\theta) = \sum_{i=1}^{n} [y_i \log h_\theta(x_i) + (1 - y_i)\log(1 - h_\theta(x_i))] + \frac{1}{2}\lambda\|\theta\|^2,$$

where $(x_i, y_i) \in \mathbb{R}^d \times \{0, 1\}$ are the data, and the classifier $h_\theta(x)$ is given by

$$h_\theta(x) = \frac{1}{1 + \exp\{-\theta^\intercal x\}}.$$

We compute the gradient and Hessian of the loss:

$$\nabla L(\theta) = \sum_{i=1}^{n} (h_\theta(x_i) - y_i)x_i + \lambda\theta = \bar{X}^\intercal(\bar{h}_\theta - \bar{Y}) + \lambda\theta \tag{22}$$

$$\nabla^2 L(\theta) = \sum_{i=1}^{n} h_\theta(x_i)(1 - h_\theta(x_i))x_i x_i^\intercal + \lambda I = \bar{X}^\intercal \bar{S}_\theta \bar{X} + \lambda I, \tag{23}$$

where $\bar{X} \in \mathbb{R}^{n\times d}$ is the data matrix whose rows are $x_i^\intercal$, $\bar{h}_\theta \in \mathbb{R}^n$ is the vector of model predictions, $\bar{Y} \in \mathbb{R}^n$ is the vector of labels, and $\bar{S}_\theta = \operatorname{diag}(\{h_\theta(x_i)(1 - h_\theta(x_i)\}_{i=1}^n) \in \mathbb{R}^{n\times n}$. Using these formulas, we can compute a Newton step for the LKO loss when we start at the minimizer θ^{full} for the full loss.

Now let $X = (x_{k+1} \cdots x_n)^\intercal \in \mathbb{R}^{(n-k)\times d}$, $Y = (y_{k+1}, \ldots, y_n)^\intercal \in \mathbb{R}^{n-k}$, $h_{\theta^{\text{full}}} = (h_{\theta^{\text{full}}}(x_{k+1}), \ldots, h_{\theta^{\text{full}}}(x_n))^\intercal \in \mathbb{R}^{n-k}$, $S_{\theta^{\text{full}}} = \operatorname{diag}(\{h_{\theta^{\text{full}}}(x_i)(1 - h_{\theta^{\text{full}}}(x_i)\}_{i=k+1}^n)$ be the LKO quantities corresponding to the terms defined above. By definition, we have

$$\begin{aligned}
\theta^{\text{Newton}} &= \theta^{\text{full}} - [\nabla^2 L^{\setminus k}(\theta^{\text{full}})]^{-1}\nabla L_{LKO}(\theta^{\text{full}}) \\
&= \theta^{\text{full}} + (X^\intercal S_{\theta^{\text{full}}} X + \lambda I)^{-1}(X^\intercal(Y - h_{\theta^{\text{full}}}) - \lambda\theta^{\text{full}}) \\
&= (X^\intercal S_{\theta^{\text{full}}} X + \lambda I)^{-1} X^\intercal S_{\theta^{\text{full}}}(X\theta^{\text{full}} + S_{\theta^{\text{full}}}^{-1}(Y - h_{\theta^{\text{full}}})) \\
&= (X^\intercal S_{\theta^{\text{full}}} X + \lambda I)^{-1} X^\intercal S_{\theta^{\text{full}}} Z,
\end{aligned} \tag{24}$$

where $Z \equiv X\theta^{\text{full}} + S_{\theta^{\text{full}}}^{-1}(Y - h_{\theta^{\text{full}}})$. Observe that equation (24) is the solution to the LKO weighted least squares problem

$$\min_{\theta} \sum_{i=k+1}^{n} h_{\theta^{\text{full}}}(x_i)(1 - h_{\theta^{\text{full}}}(x_i))(\theta^\intercal x_i - z_i)^2 + \lambda\|\theta\|^2, \tag{25}$$

where z_i is the i-th component of $\bar{Z} \equiv \bar{X}\theta^{\text{full}} + \bar{S}_{\theta^{\text{full}}}^{-1}(\bar{Y} - \bar{h}_{\theta^{\text{full}}})$. By adapting the PRU to this situation, we can compute a fast approximation to the Newton step.

We can compute the vector $\bar{Z} \equiv \bar{X}\theta^{\text{full}} + \bar{S}_{\theta^{\text{full}}}^{-1}(\bar{Y} - \bar{h}_{\theta^{\text{full}}})$, as well as the matrix $H_{\lambda,\bar{h}_{\theta^{\text{full}}}} \equiv \bar{X}(\bar{X}^\intercal \bar{S}_{\theta^{\text{full}}}\bar{X} + \lambda I)^{-1}\bar{X}^\intercal \bar{S}_{\theta^{\text{full}}}$, offline. Observe that $H_{\lambda,\bar{h}_{\theta^{\text{full}}}}$ is the hat matrix for the "full" least squares problem

$$\min_{\theta} \sum_{i=1}^{n} h_{\theta^{\text{full}}}(x_i)(1 - h_{\theta^{\text{full}}}(x_i))(\theta^\intercal x_i - z_i)^2 + \lambda\|\theta\|^2. \tag{26}$$

For consistency with the rest of the paper, let $\theta^{\setminus k}$ be the exact solution to (25) (so $\theta^{\setminus k} = \theta^{\text{Newton}}$). By the result of Theorem 4, we can compute the LKO model predictions $\hat{z}_i^{\setminus k} \equiv \theta^{\setminus k\intercal} x_i$, $i = 1, \ldots, k$ in $O(k^3)$ time. Observe that the gradient of the (unregularized, unweighted, quadratic) loss on the synthetic points $(x_i, \hat{z}_i^{\setminus k})$ is

$$\sum_{i=1}^{k} (\theta^{\text{full}\intercal} x_i - \hat{z}^{\setminus k}) x_i = \left(\sum_{i=1}^{k} x_i x_i^\intercal\right)(\theta^{\text{full}} - \theta^{\setminus k}). \tag{27}$$

We are now in a setting exactly analogous to equation (4), even though θ^{full} was the minimizer for the original *cross-entropy* objective rather than (26). By mimicking the proof of Theorem 1 from this point, we can derive the exact same results. Namely, the step taken by the projective residual update is equal to $\text{proj}_{\text{span}(x_1,\ldots,x_k)}(\theta^{\setminus k} - \theta^{\text{full}})$. By definition of $\theta^{\setminus k}$ and of the Newton step, it follows that $\theta^{\setminus k} - \theta^{\text{full}} = \Delta_{\text{Newton}}$. Combining these two facts yields the statement of Theorem 6. The computational cost calculation is identical to the calculation in Theorem 1. □

F Synthetic data construction

We first generate a matrix of n d-dimensional covariates $X \in \mathbb{R}^{n\times d}$; we do this by drawing the rows $x_i^\intercal$ of X according to $x_i \overset{\text{i.i.d.}}{\sim} N(0, \Sigma)$, where Σ is randomly selected via `sklearn.datasets.make_spd_matrix` (Pedregosa et al., 2011). Once X is generated, the response vector $Y \in \mathbb{R}^n$ is generated by randomly selecting a (fixed) "true" underlying parameter $\theta^* \in \mathbb{R}^d$, and setting $Y = X\theta^* + \varepsilon$, where $\varepsilon \sim N(0, \sigma^2 I_n)$ is the error vector. For our experiments, we set the noise level $\sigma^2 = 1$; for reasonable values of σ^2 this parameter does not play a large role in the outcome of the experiments. For all of the synthetic experiments, when deleting a group of size k, we always assume that it is the first k datapoints which are being deleted. (That is, we delete the datapoints specified by the first k rows of X and the first k entries of Y.) For all of the synthetic datasets, we take $n = 10d$.

For the runtime experiment, no modifications are made to the general setup. We vary the dimension d between $d = 1000$ and $d = 3000$ and the group size k between $k = 1$ and $k = 100$.

For the L^2 experiment, we first construct $\tilde{X}$ and $\tilde{Y}$ according to the general procedure above. We then obtain the data X, Y by multiplying the first k rows of $\tilde{X}$ and the first k entries of $\tilde{Y}$ (that is, the points which will eventually be deleted) by a factor λ to demonstrate the effectiveness of each method at removing outlier datapoints. This is the setting described in section 4.1.

The modifications for the FIT experiment are slightly more involved. We construct sparse data with three key properties: (1) only the deleted feature vectors x_i, $i = 1, \ldots, k$ have nonzero d-th entry (this is the "injected feature"); (2) the deleted feature vectors all lie on the same low-dimensional subspace; (3) the response for the deleted points is perfectly correlated with the special feature. The exact steps for this procedure are as follows:

1. Construct $\tilde{X}$ according to the general procedure. (Pick a random covariance Σ and draw the rows $x_i^\intercal$ of $\tilde{X}$ according to $x_i \overset{\text{i.i.d.}}{\sim} N(0, \Sigma)$.)

2. The "injected feature" will be the last (d-th) entry of each vector x_i. Since only the group being removed has the injected feature, we set the last entry in rows $k+1$ to n of $\tilde{X}$ equal to 0; the first k rows keep their original final entry.

3. Sparsify $\tilde{X}$ so that it has a fraction of approximately p nonzero entries. Let $\tilde{X}[i,j]$ denote the (i,j)-th entry of $\tilde{X}$.

   (a) Sparsify the first k rows of $\tilde{X}$ simultaneously: for each $j = 1, \dots, d-1$, set $\tilde{X}[i,j] = 0$ for all $i = 1, \dots, k$ with probability $1-p$.

   (b) Sparsify the remaining entries of $\tilde{X}$: for each $i = k+1, \dots, n$ and $j = 1, \dots, d-1$, set $X[i,j] = 0$ with probability $1-p$.

4. Let X be the matrix resulting from performing operations 1-3 on $\tilde{X}$. Set $\tilde{Y}$ according to the general procedure: $\tilde{Y} = X\theta^* + \varepsilon$.

5. Let $\tilde{Y}[i]$ denote the i-th entry of $\tilde{Y}$. For $i = 1, \dots, k$, set $\tilde{Y}[i] = w_* X[i,d]$, where w_* is the pre-specified "true" weight of the injected feature.

For the sparse logistic regression experiment, the matrix of covariates X is generated according to the procedure above. The labels Y are then generated so that the logistic model is well-specified, as described in section 6.2.

To generate data for the FIT for logistic regression, the procedure is the same as the one outlined for linear regression above (and as outlined in Section 5) with some minor changes. WLOG assume that the points to be deleted are $(X[i,:], Y[i])_{i=1}^k$. We require the following:

1. The k points to be deleted all belong to the positive class, i.e. $Y[i] = 1$, $i = 1, \dots, k$.

2. The k points to be deleted are classified correctly by the full model, i.e. $X[i,:]^\intercal \theta^{\text{full}} > 0$, $i = 1, \dots, k$. ($X[i,:]$ denotes the i-th row of the data matrix X.)

3. The k points to be deleted have injected feature equal to 1, while the points that remain all have injected feature equal to 0. That is, $X[i,d] = 1$ for $i = 1, \dots, k$ and $X[i,d] = 0$ for all $i > k$.

G Baseline values for synthetic linear regression experiments

All of the experimental results in the main body of the paper are given relative to an absolute baseline value. In Tables 7, 8, and 9, we report the medians of the absolute baseline values to which we are comparing for each of the three synthetic experiments (runtime, L^2, and feature injection, respectively) for linear regression. The baseline values follow the trends we would expect. In particular, the runtimes increase sharply as the dimension increases and slowly as the group size increases (Table 7); the unique feature weight originally learned by the model is close to its true value, 10 (Table 8); and the distance between θ^{full} and $\theta^{\setminus k}$ increases with the number of points removed, as well as the dissimilarity of these points to the rest of the dataset as measured by the multiplier λ (Table 9).

Table 7: Median exact retraining runtimes in seconds for table 2. The method used was a Newton step with the Sherman-Morrison formula.

	$d = 1000$	$d = 1500$	$d = 2000$	$d = 2500$	$d = 3000$
$k = 1$	0.08	0.27	0.67	1.19	2.25
$k = 5$	0.08	0.31	0.67	1.33	2.22
$k = 10$	0.08	0.31	0.63	1.33	2.04
$k = 25$	0.08	0.32	0.62	1.36	2.06
$k = 50$	0.09	0.33	0.64	1.40	2.11

Table 8: Median baseline weights on injected feature for table 3.

	$p = 0.25$	0.1	0.05
$k = 5$	8.97	10.61	11.03
$k = 50$	10.23	9.73	10.10
$k = 100$	9.51	9.99	10.01

Table 9: Median baseline L^2 parameter distances for table 4.

	$\lambda = 1$	$\lambda = 10$	$\lambda = 100$
$k = 5$	0.018	0.175	0.192
$k = 50$	0.057	0.523	0.572
$k = 100$	0.082	0.761	0.842

H Detailed experimental results for logistic regression

Here we give the complete results—inlcuding the results for Newton's method, as well as the IQR for each setting—for the logistic regression experiments. As explained in appendix E, the logistic PRU computes a projection of the Newton step onto a lower-dimensional subspace. (In fact, for both linear and logistic regression, the PRU computes a projection of the Newton step. It just happens that for linear regression, the Newton step is exact, while this is no longer the case for logistic regression.) As a result, retraining via Newton's method is more accurate than retraining via the PRU. The PRU's advantage lies in its combination of accuracy and speed. While slightly less accurate than Newton's method, the PRU can be up to thousands of time faster. Indeed, since the computational cost of Newton's method for logistic regression is the same as the computational cost of exact retraining for linear regression, the PRU has the same favorable runtime comparisons as in Table 2.

Table 10: Complete results for the sparse logistic FIT. For larger group sizes and sparse data, the PRU is able to completely remove the injected feature. With any strictly positive regularization, Newton's method will completely remove the injected feature, but its computational cost is vastly slower than that of the PRU (see Table 2).

	$p = 0.5$	$p = 0.1$	$p = 0.05$
$k = 25$ (INF)	**0.82 (0.79 - 0.82)**	0.76 (0.73 - 0.80)	0.78 (0.77 - 0.79)
$k = 25$ (PRU)	0.86 (0.83 - 0.88)	**0.69 (0.64 - 0.70)**	**0.44 (0.40 - 0.50)**
$k = 25$ (NWT)	0.0 (0.0 - 0.0)	0.0 (0.0 - 0.0)	0.0 (0.0 - 0.0)
$k = 50$ (INF)	0.81 (0.78 - 0.84)	0.82 (0.81 - 0.83)	0.82 (0.80 - 0.84)
$k = 50$ (PRU)	0.81 (0.78 - 0.84)	**0.48 (0.48 - 0.54)**	**0.02 (0.00 - 0.03)**
$k = 50$ (NWT)	0.0 (0.0 - 0.0)	0.0 (0.0 - 0.0)	0.0 (0.0 - 0.0)
$k = 100$ (INF)	0.82 (0.81 - 0.83)	0.85 (0.83 - 0.86)	0.84 (0.82 - 0.85)
$k = 100$ (PRU)	**0.71 (0.69 - 0.71)**	**0.00 (0.00 - 0.01)**	**0.0 (0.0 - 0.0)**
$k = 100$ (NWT)	0.0 (0.0 - 0.0)	0.0 (0.0 - 0.0)	0.0 (0.0 - 0.0)

Table 11: Complete results for the sparse logistic L^2 experiment. For sparse data and moderate group deletion sizes, the PRU's performance surpasses the performance of the influence method. The PRU becomes nearly as accurate as Newton's method while maintaining a faster runtime.

	$p = 0.5$	$p = 0.1$	$p = 0.05$
$k = 25$ (INF)	**0.85 (0.81 - 0.87)**	**0.78 (0.75 - 0.80)**	0.78 (0.77 - 0.79)
$k = 25$ (PRU)	0.86 (0.84 - 0.87)	0.80 (0.79 - 0.81)	**0.65 (0.63 - 0.69)**
$k = 25$ (NWT)	0.08 (0.07 - 0.09)	0.02 (0.02 - 0.03)	0.01 (0.01 - 0.02)
$k = 50$ (INF)	0.85 (0.82 - 0.86)	0.83 (0.81 - 0.83)	0.82 (0.80 - 0.84)
$k = 50$ (PRU)	0.85 (0.83 - 0.86)	**0.69 (0.68 - 0.72)**	**0.20 (0.18 - 0.25)**
$k = 50$ (NWT)	0.08 (0.07 - 0.09)	0.03 (0.02 - 0.03)	0.01 (0.01 - 0.02)
$k = 100$ (INF)	0.85 (0.84 - 0.85)	0.86 (0.84 - 0.87)	0.84 (0.82 - 0.85)
$k = 100$ (PRU)	**0.80 (0.79 - 0.81)**	**0.24 (0.21 - 0.24)**	**0.13 (0.12 - 0.14)**
$k = 100$ (NWT)	0.09 (0.08 - 0.09)	0.03 (0.02 - 0.04)	0.01 (0.01 - 0.01)